BIULETYN INSTYTUTU AUTOMATYKI I ROBOTYKI
NR 21, 2004

Testy penetracyjne w badaniu skuteczności IDS dla środowiska Unix/Linux

Adam KLEPKA, Zbigniew SUSKI

Instytut Teleinformatyki i Automatyki WAT
ul. Kaliskiego 2, 00-908 Warszawa

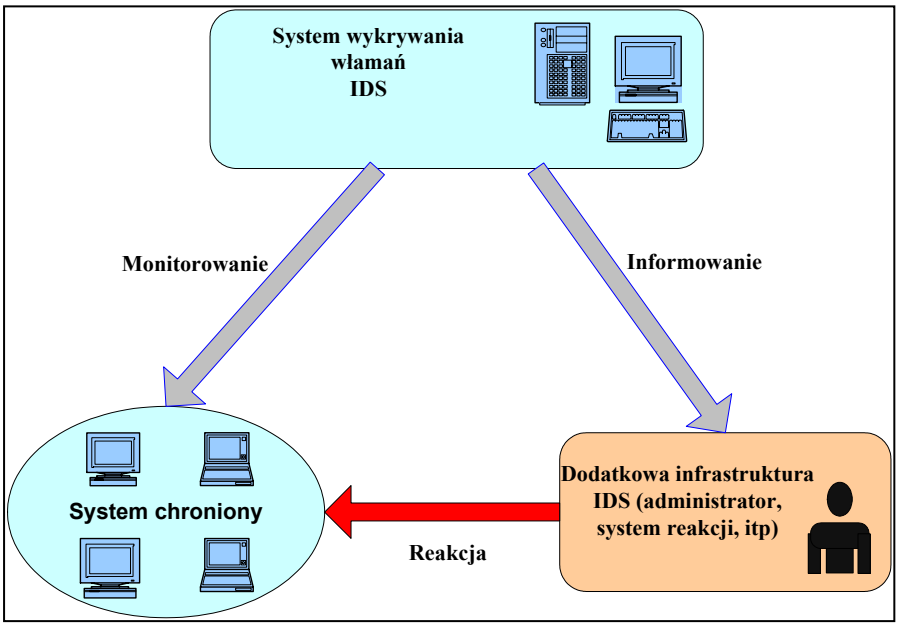
STRESZCZENIE: W artykule opisany został eksperyment, którego celem było zbadanie, poprzez wykorzystanie testu penetracyjnego, skuteczności ochrony sieci przez wybrane produkty typu IDS dla środowiska Unix/Linux. Projekt powstał w ramach pracy dyplomowej, której celem było dokonanie analizy porównawczej wybranych systemów wykrywania włamań dla środowiska Unix/Linux.

1. Wstęp

Wykrywanie włamań jest „procesem monitorowania zdarzeń zachodzących w systemie komputerowym lub sieci, ich analizy pod kątem zawartości instrukcji mających na celu kompromitację atrybutów bezpieczeństwa: poufności, spójności, dostępności lub ominięcia mechanizmów zabezpieczających system lub sieć komputerową”[*]. Inaczej, można powiedzieć iż jest to „proces identyfikowania i reagowania na szkodliwą działalność skierowaną przeciw zasobom informatycznym” [1]. Systemy wykrywania włamań realizują trzy podstawowe zadania:

- monitorowanie i kontrola zdarzeń,
- informowanie administratorów,
- reagowanie na występujące zdarzenia.

[*] National Institute of Standards and Technology “Special Publication on Intrusion Detection System”- R.Bace, P. Mell



Rys. 1. Ogólna architektura systemu wykrywania włamań

2. Przegląd typowych ataków na środowisko UNIX/Linux.

Włamanie jest rodzajem ataku obejmującym wszelkie działania intruza, które powodują wystąpienie zagrożeń naruszenia bezpieczeństwa zasobów teleinformatycznych. W ramach podstawowych atrybutów bezpieczeństwa informacji możemy wyróżnić:

- poufność – informacje przesyłane w sieci i składowane nie mogą być czytane przez nieuprawnione podmioty,
- integralność – elementy sieci oraz informacje nie mogą zostać zniszczone, uszkodzone lub skradzione, nie mogą też zostać zmodyfikowane,
- dostępność – elementy sieci, dane i usługi są dostępne zawsze, kiedy są potrzebne użytkownikom

Czynności jakie potencjalny intruz wykonuje przed dokonaniem naruszenia polityki bezpieczeństwa w systemie często nazywane są modelowaniem włamań. W skład tych działań wchodzi proces zbierania informacji o potencjalnym celu np. strukturze sieci, na którą ma nastąpić atak, pozostawienie lub odnalezienie słabego punktu sieci lub oprogramowania

podatnego na włamanie, zabezpieczenie usunięcia wszelkich śladów bytności intruza, itp.

Do podstawowych rodzajów ataków można zaliczyć:

- Nieautoryzowany dostęp do zasobów:
  - łamanie haseł i praw dostępu,
  - konie trojańskie,
  - uprowadzenia - zwykle są to uprowadzenia połączeń TCP/IP często wymagające dodatkowych mechanizmów dla zablokowania działania właściwego systemu (np. poprzez powódź), ataki typu MITM (man in the middle),
  - podszywanie ,
  - skanowanie portów i aktywności systemu,
  - zdalne wykrywanie rodzaju i wersji systemu operacyjnego (fingerprint),
  - podsłuchiwanie,
  - kradzieże danych, np. zbiorów,
  - nadużycia legalnego dostępu,
  - nieupoważnione podłączenia do sieci,
  - wykorzystanie zasobów systemu do prywatnych celów,
  - wykorzystanie luk systemowych do pozyskania zasobów lub zdobycia uprawnień.
- Nieuprawnione zmiany zasobów (następujące po uzyskaniu dostępu):
  - modyfikacje uprawnień, np. nadanie sobie praw administratora,
  - modyfikacje i kasowanie danych,
  - nieuprawnione transmisje i tworzenie danych,
  - nieupoważniona konfiguracja (modyfikacja) systemów i usług sieciowych (serwerów).
- Blokowanie usług (Denial of Service)
  - Powodzie:
    - ping flood (Smurf),
    - pocztowe (mail flood),

- SYN flood,
- powodzie rozproszone - DDoS (Distributed Denial of Service).

- Wyłączenia systemów poprzez wykorzystanie ich luk, ataki na aplikacje:
  - przepełnienia buforów, np. Ping of Death,
  - preparowanie pakietów, np. *chargen*, *teardrop*, *land*,
  - zdalne wyłączenie.

3. Opis badanych systemów wykrywania włamań

3.1. SNORT

Snort jest pojedynczym procesem podsłuchującym pakiety dostępne na wskazanych interfejsach. Można go zresztą wykorzystać w charakterze klasycznego sniffera takiego jak np. *tcpdump*. Podstawą obu narzędzi jest biblioteka *libpcap*, stanowiąca praktyczny standard interfejsu dla programów przechwytujących i filtrujących pakiety sieciowe. Dodatkowo Snort ma możliwość zapisywania przechwyconych pakietów w różnych formatach, co pozwala na wykorzystanie go jako czujnika zbierającego dane i przekazującego je do dalszej analizy. Kolejnym, kluczowym elementem Snorta są moduły analizowania zebranych informacji.

Preprocesory

Jak już powiedziano, do podstawowych funkcji Snort'a należy przechwytywanie i zapisywanie pakietów. O tym, jakie pakiety będą zapisywane i ewentualnie wzbudzą alarm decydują tzw. preprocesory, włączane w konfiguracji Snort'a. Moduły te analizują różne cechy charakterystyczne pakietów, mogąc wykrywać najrozmaitsze ataki. Preprocesory mogą również przygotowywać strumień danych dla kolejnych modułów IDS. Do podstawowych preprocesorów Snort'a należą:

- *http_decode* - przetwarza adresy URL, konwertując na ASCII znaki przedstawione w postaci szesnastkowej. Ułatwia to wykrycie ataku, w którym ukrywa się sygnatury ataków przez zakodowanie typowych sekwencji,

- *portscan* - wykrywa próby skanowania portów, identyfikowane przez przekroczenie progowej liczby prób połączeń, w określonym przedziale czasu,
- *frag2* – pozwala na składanie danych przychodzących w postaci fragmentów, co umożliwia wykrywanie ataków prowadzonych za pomocą framentacji pakietów,
- *stream4* - analizuje strumienie TCP. Wykrywa niektóre formy skanowania portów, sprawdza poprawność stanów połączeń oraz sum kontrolnych pakietów (TCP, UDP i ICMP),
- *unidecode* - dekoduje znaki z formatu Unicode, oraz wykrywa nielegalne kodowania,
- *rpc_decode* - normalizuje żądania wykorzystujące protokół RPC, umożliwiając wykrywanie podejrzanych pakietów za pomocą operacji cząstkowych,
- *telnet_decode* - usuwa z sesji TELNET i FTP binarne ciągi mogące utrudnić wyszukiwanie sygnatur ataków,
- *bo* - wyszukuje w pakietach UDP próby połączeń konia trojańskiego o nazwie *Back Orifice*,
- *arpspoof* - wykrywa podejrzane pakiety ARP, mogące sygnalizować próby spoofingu ARP.

Sygnatury

Kolejnym etapem weryfikacyjnym przez jaki przechodzi pakiet jest moduł wykrywania ataków metodą analizy sygnatur. Reguły tworzące sygnatury, mogą odnosić się do różnych charakterystycznych cech pakietów. Najczęściej wykorzystywanymi są:

- adresy IP,
- numery portów,
- parametry nagłówka IP (TTL, TOS, opcje, fragmentacja, długość),
- parametry nagłówka TCP (flagi, numery sekwencyjne, wartości ACK),
- rodzaje i typy pakietów ICMP,
- łańcuchy tekstowe i binarne występujące w pakietach.

Reagowanie na ataki

Do podstawowych sposobów reagowania na ataki można zaliczyć:

- zgłaszanie alarmów,
- zapisanie zawartości do pliku,
- przerwanie połączenia TCP,
- wywołanie zewnętrznych poleceń (np. rekonfiguracja zapory sieciowej i zablokowanie dostępu z sieci).

Alerty

Możliwe są następujące sposoby przekazywania komunikatów alarmowych:

- zarejestrowanie w dzienniku systemowych (*syslog*),
- zarejestrowanie w pliku w postaci *fast* (pojedyńcze wiersze z podstawowymi informacjami) lub *full* (pełne informacje o zdarzeniu plus zawartość pakietu zapisana do odpowiedniego katalogu),
- wyświetlenie okienka *WinPopup* z alertem na stacji roboczej administratora za pomocą protokołu SMB,
- wysłanie alertu na „gniazdo uniksowe" (*unix socket*), co pozwala na wykorzystanie zewnętrznego programu do dalszej analizy i podjęcia działania,
- zapisanie przechwyconego pakietu w binarnym formacie *tcpdump*,
- wysłanie pułapki SNMP do wskazanego serwera,
- zapisanie rekordu w formacie CSV, czyli tekstu z separatorami nadającego się do łatwego importu do bazy danych,
- zapisywanie kompletu danych w formacie *unified*, czyli osobno danych i zawartości pakietów, w postaci nadającej się do późniejszej analizy przez zewnętrzny program,
- zapisanie rekordu do lokalnej lub zewnętrznej bazy danych przez odpowiednie ODBC (Snort obsługuje bazy MySQL, PostgreSQL i Oracle).

3.2. Prelude IDS

Architektura

Prelude IDS jest hybrydowym systemem wykrywania włamań, który potrafi wykrywać i monitorować nieprawidłowości na poziomie ruchu sieciowego jak również na poziomie kronik systemu operacyjnego. Składa się z pięciu głównych modułów:

- *Prelude-manager* – moduł, który zbiera informacje przesyłane od sensorów. Po otrzymaniu sygnału od sensora, moduł ten wpisuje przesłane dane do pliku lub bazy danych *MySql*. W ten sposób użytkownik uzyskuje możliwość analizy zapisanych informacji w jednym centralnym miejscu.
- *Prelude-nids* –moduł analizujący ruch pakietów w sieci. Dostarcza funkcjonalności podobnej do sensorów Snorta. Jest używany w celu wykrywania incydentów i przesyłania informacji do modułu *prelude-manager*.
- *Prelude-lml* - moduł analizy anomalii na poziomie logów systemu operacyjnego. Jego działanie polega na analizie określonych logów w poszukiwaniu zdefiniowanych zdarzeń. Po wykryciu anomalii wysyłany jest komunikat do modułu *prelude-manager*.
- *Libprelude* – biblioteka, która dostarcza niezbędnych funkcji komunikacji pomiędzy modułami oraz funkcji dla sensorów.
- *Piwi* –interfejs użytkownika umożliwiający prezentację alertów, modyfikację i zarządzanie regułami oraz raportowanie. Moduł ten został całkowicie napisany w *perlu*.

Zarówno sensor jak i *prelude-manager* posiadają możliwość zapisania informacji o występujących zdarzeniach na komputerze lokalnym. W przypadku braku połączenia pomiędzy sensorami i punktami dystrybucji komunikatów informacje takie są zapisywane lokalnie, a po przywróceniu komunikacji przesyłane są do centralnego modułu zarządzającego. Każdy z tych modułów używa biblioteki *libprelude*, która zapewnia poprawną komunikację pomiędzy wszystkimi agentami Prelude. Biblioteka ta może także zostać wykorzystana jako interfejs dla innych programów, w celu zapewnienia kompatybilności w procesie komunikacji.

Architektura komponentów Prelude zbudowana jest w taki sposób, aby zapewnić możliwość budowania sieci sensorów (*Prelude-nids, Prelude-lml*). Sensory wysyłają alerty o zauważonych incydentach do modułu zarządzającego, który z kolei jest powiązany z interfejsem użytkownika (*piwi*). Interfejs *piwi*

dostarcza graficznej reprezentacji otrzymanych alertów, pozwala na zarządzanie regułami, filtrowanie występujących zdarzeń oraz raportowanie.

Sensor

Sensor Prelude jest odpowiedzialny za wykrywanie incydentów zachodzących w sieci lub na serwerze w czasie rzeczywistym. Może zostać zbudowany jako struktura bardzo złożona lub pracować jako pojedynczy proces, który sprawdza czy na danych serwerach funkcjonują otwarte określone porty. Proces komunikacji z modułem zarządzającym odbywa się z wykorzystaniem biblioteki *libprelude*.

Podstawowym sensorem sieciowym jest *prelude-nids*. Dokonuje on analizy w czasie rzeczywistym przechwytywanych pakietów. Do tego celu używa zmodyfikowanej wersji biblioteki *pcap*. Po przechwyceniu pakietu jego nagłówek jest dekodowany i zapisywany w wewnętrznej strukturze danych. Następnie wykonuje się szereg testów i porównań aby określić czy pakiet może zostać uznany za właściwy czy niepoprawny – np. weryfikuje się poprawną budowę pakietu. W kolejnym zestawie testów bada się flagi nagłówka TCP oraz IP. W przypadku wykrycia anomalii pakiet jest odrzucany aby zapobiec błędom w dalszej analizie.

Następnym etapem testów jest weryfikacja fragmentacji pakietu. Pakiet z ustawioną flagą MF (*more fragments*) jest składany przez sensor i analizowany. Analiza jest dokonywana po otrzymaniu wszystkich części pakietu.

Preprocesory

Podstawowe preprocesory Prelude-IDS są wykorzystywane w fazie analizy wyższych warstw protokołów. Zostały one przedstawione poniżej.

- *HTTP Decoding Plugin* – moduł dekodowania zapytań HTTP. Przeszukuje żądania HTTP w poszukiwaniu nieprawidłowych znaków kodowych. Potrafi wykryć pewną liczbę ataków związanych z nieprawidłową sekwencją kodowania UTF-8 lub ukrywaniem znaków.
- *Escaped Characters Decoding in FTP and TELNET* – moduł ten wykrywa opcje pakietów w sesji telnet i ftp. Próbuje normalizować opcje pakietów w celu uniknięcia niepożądanych sytuacji zakłócających normalną komunikację.
- *RPC Decoding* – moduł dekodujący pakiety UDP oraz TCP zawierające nagłówki RPC. Poddawane są one analizie sygnatur i testowane pod kątem właściwości rpc.

- *Plugins' Job* – monitoruje i analizuje specyficzne protokoły, dla których został zaprojektowany - protokoły, dla których nie jest możliwa analiza sygnatur zawartości pakietów.
- *SnortRules Plugin* – oferuje mechanizmy kompatybilności dla sygnatur Snorta.
- *Scan Detection Plugin* – generuje alerty w przypadku wykrycia dużej liczby połączeń do różnych portów w przeciągu określonego czasu.
- *Polymorphic Shell Code Detection Plugin* – wyszukuje sekwencje wszystkich kodów operacji instrukcji NOP (hex. 0x90 na architekturze 32-bit). Instrukcja ta jest wykorzystywana do ataków związanych z przepełnieniem bufora. W przypadku stwierdzenia wystąpienia określonej ilości instrukcji NOP w ciągu pakietu generowany jest alert.
- *ArpSpoof Plugin* – sprawdza związek logiczny pomiędzy wiadomością ARP i nagłówkiem ramki ethernet. Alerty generowane są w przypadku:
  - żądanie ARP jest wysyłane na adres UNICAST,
  - adres źródła ramki ethernetowej jest różny od adresu źródła w żądaniu ARP,
  - zostaje wykryty konflikt w bazie „*arpwatch*", która pozwala na przypisanie adresów IP do adresów sprzętowych MAC
- *IP Defragmentation stack* – zapobiega wysłaniu pakietów przekraczających określony przez MTU rozmiar. Atakujący może użyć fragmentacji IP do ukrycia przed sensorem określonego ciągu formatującego, zawierającego potencjalnie niedozwolone instrukcje.
- *TCP stream reassembly* – wykonuje ponowne złożenie strumienia TCP, w celu jego analizy, Złożenie strumienia TCP pozwala na dokładniejszą analizę, dostrzeżenie i powstrzymanie ataków, w których używane są techniki omijania systemów wykrywania włamań.

Alerty

Komunikaty o wykrytych incydentach są przesyłane przez sensory do modułu zarządzającego. Tam poddawane są analizie i przesyłane do podsystemu nazwanego „*Counter Measure subsystem*", który może zostać skonfigurowany do określonej reakcji na występujące zdarzenia. Jeśli moduł reakcji określi do wykonania czynność zapobiegającą atakowi, rozkaz wykonania polecenia jest wysyłany do zdalnych agentów. Prelude IDS posiada możliwość zapisywania alertów w następujących formatach:

- baza danych MySQL,
- baza danych PostgreSQL,
- baza danych Oracle,
- format XML,
- format tekstowy.

4. Metodyka przeprowadzania testów

Metodyka to zbiór zasad i sposobów dotyczących wykonywania jakiejś pracy i zmierzania do określonych celów”[*]. Do podstawowych elementów składowych każdej metodyki można zaliczyć:

- używane narzędzia, opis środowiska,
- opis sposobu realizacji badania,
- sposób dokumentowania przeprowadzanych badań,
- kryteria oceny.

4.1. Cel badania

Celem badania jest doświadczalne określenie i porównanie jakości zabezpieczeń oferowanych przez wybrane systemy wykrywania włamań dla środowiska Unix. Do istotnych wyznaczników jakości można zaliczyć możliwości i skuteczność zabezpieczeń, zarządzanie bezpieczeństwem, a także certyfikaty i testy jakości wykonywane przez zewnętrzne instytucje. W zakresie możliwości i skuteczności zabezpieczeń IDS istotne są następujące własności:

- wykrywalność ataków - IDS identyfikuje ataki, dla których posiada zdefiniowane sygnatury i jest odporny na techniki oszukiwania zabezpieczeń (*evasion techniques*),
- fałszywe alarmy - IDS precyzyjnie analizuje ruch sieciowy i generuje niewielką liczbę fałszywych alarmów (*false positives*),
- wydajność zabezpieczeń - IDS nawet w warunkach dużego obciążenia sieci poddaje analizie całość ruchu sieciowego tzn. „nie gubi pakietów”,

[*] Prof. dr. Jan Tokarski – Słownik wyrazów obcych PWN

- reakcja na ataki - IDS w razie wykrycia ataku podejmuje stosowne działanie w zakresie ochrony zasobów systemu informatycznego (np. informuje administratora, zabija/blokuje sesję).

4.2. Stanowisko badawcze

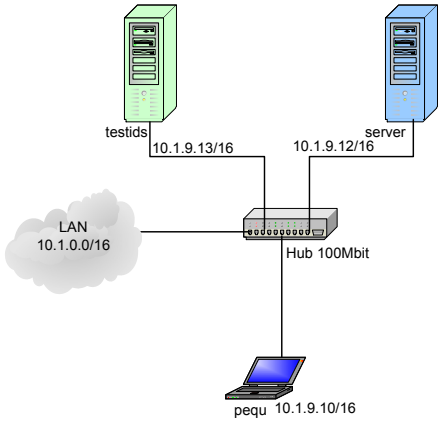
Konfiguracja sprzętowa

Testowe środowisko składa się z trzech komputerów połączonych interfejsami sieciowymi poprzez hub 100 Mb. Na każdym z komputerów zainstalowany jest system operacyjny Linux v. 2.4.22-xfs (root@Knoppix) (gcc version 2.95.4 20011002 (Debian prerelease)).

Komputery o nazwach *testids* oraz *server* wyposażone są w 256MB pamięci RAM, kartę sieciową 100Mb/s, dysk twardy typu IDE 10GB, klawiaturę. Procesor dla systemu *testids* taktowany jest częstotliwością 800MHz, dla komputera *server* 560MHz. Komputer o nazwie *pequ* posiada 512MB pamięci RAM, partycję 5GB na potrzeby instalacji systemu operacyjnego i wymaganych narzędzi, procesor taktowany częstotliwością 1.3MHz. W tabeli 1 przedstawiono nazwy komputerów, ich adresy IP oraz funkcje pełnione w środowisku testowym.

Tabela 1. Konfiguracja interfejsów sieciowych

Nazwa serwera	adres IP	Nazwa interfejsu	Funkcja
testids	10.1.9.13	eth0	Serwer z zainstalowanym systemem IDS
server	10.1.9.12	eth0	Komputer świadczący usługi w sieci dla innych użytkowników
pequ	10.1.9.10	eth0	Komputer atakujący



Rys. 2. Schemat badanego środowiska

Zainstalowane oprogramowanie

Na wszystkich komputerach instalowane było standardowe programowanie dostarczane w ramach dystrybucji. Dodatkowo na komputerze *testids*, zainstalowane zostały pakiety wymagane i związane z badanym systemem IDS. Proces instalacji i lista wymaganych pakietów dla konkretnego systemu IDS została opisana w rozdziale 3 niniejszego opracowania. Na komputerze *pequ*, zainstalowano dodatkowo:

- *tethereal* - analizator protokołów sieciowych. Pozwala na przechwytywanie pakietów bezpośrednio z sieci lub odczytanie pakietów z wcześniej zapisanego pliku. Ponadto wyświetla zdekodowaną formę pakietu umożliwiając jego analizę.
- *xprobe2* - narzędzie do rozpoznawania systemów operacyjnych. Xprobe2 używa różnorodnych metod porównywania sygnatur odpowiedzi serwera korzystając z gotowej bazy sygnatur
- *hping2* - narzędzie pozwalające na dowolne preparowanie wysyłanych pakietów sieciowych. Obsługuje protokoły: TCP, UPD, ICMP.
- *nemesis* - program pozwalający na tworzenie i modyfikowanie pakietów. Jest dobrym narzędziem do testowania systemów IDS, zapór sieciowych i innych. Umożliwia zbudowanie pakietów ARP, DNS, ETHERNET, ICMP, IGMP, IP, OSPF, RIP, TCP i UDP.
- *fragroute, fragtest* - przechwytuje, modyfikuje i wysyła pakiety do określonego serwera. Udostępnia następujące mechanizmy ataków – duplikowanie pakietu, utracenie pakietu, fragmentacja pakietu, zachodzenie pakietów, zmiana porządku.
- *nikto*- skaner sieciowy dla usług www. Testuje serwisy www pod kątem znanych podatności z wykorzystaniem skryptów cgi,
- *apache_exploit* –program wykonujący atak na serwer *apache*,
- *samba_exploit* –program wykonujący atak na serwer samby,
- *wuftpd_exploit* –program wykonujący atak na usługę ftp,
- *IDSwakeup* –program wykonujący szereg popularnych ataków. Posiada mechanizmy omijania systemów IDS.

Na komputerze *server* z uruchomionymi usługami www, mysql, ftp dodatkowo zainstalowano: serwer Apache, serwer MySql, parser języka php.

Potrzebne dokumenty

Przed badaniem powinny zostać przygotowane następujące dokumenty:

- plan połączeń sieci testowej, identyfikacja serwerów, numeracja IP,
- opis wykorzystanych narzędzi, poleceń do przeprowadzenia badania,
- opis oprogramowania testowanego (IDS), jego konfiguracja, oraz oprogramowania związanego z systemem testowanym,
- opis ataku. Nie wymaga się aby przeprowadzany atak zakończony był powodzeniem.

Przygotowanie środowiska do badań

W ramach przygotowania środowiska do badań należy wykonać następujące czynności:

- sprawdzenie konfiguracji posiadanych zestawów komputerowych,
- instalacja właściwego systemu operacyjnego na wszystkich wymaganych komputerach według przyjętych założeń,
- instalacja i konfiguracja wymaganych usług,
- instalacja i konfiguracja systemu wykrywania włamań.

4.3. Procedura testowa

Procedura testowa rozpoczyna się od przeprowadzenia szeregu działań mających na celu identyfikację rodzaju i wersji systemu operacyjnego (*OS fingerprint*), np. poprzez badanie typowych reakcji na pakiety o określonych cechach. Program xprobe skanuje otwarty port serwera i na podstawie standardowych odpowiedzi usług typowych dla danego systemu pozwala z dużym prawdopodobieństwem określić typ i wersję systemu operacyjnego. Następnie przeprowadzana jest wstępna penetracja systemu za pomocą skanerów portów TCP i UDP z wykorzystaniem różnych technik skanowania dostarczanych przez program nmap. Kolejnym krokiem są testy polegające na modyfikacji pakietów wysyłanych do określonego serwera w środowisku testowym. W ramach tych testów przeprowadzane są ataki typu smurf oraz teardrop. W dalszej kolejności przeprowadzany jest atak oparty na fałszowaniu tablicy ARP określonych serwerów. Do tego celu zostało wykorzystane polecenie arpspoof z odpowiednimi parametrami. W kolejnym kroku badań przeprowadzane są ataki z grupy DoS. Atak MAC flood przeprowadzany jest z wykorzystaniem narzędzia ettercap. Ostatni zestaw ataków przeprowadzany

jest na dostępne usługi: www, mysql, ftp. Do przeprowadzenia ataków używane są polecenia systemu operacyjnego oraz formaty poleceń przeglądarki www. Testy systemu wykrywania włamań kończą się uruchomieniem programów napisanych w perlu, które testują najczęściej występujące podatności w serwerach www, oraz programów skompilowanych na potrzeby testu: *apache_exploit*, *samba_exploit*, *wuftpd_exploit* oraz I*DSwakeup*.

W trakcie badań sporządzane są notatki z przebiegu ataku opisujące m.in. zaobserwowane działanie systemu wykrywania włamań. Analizie zostają poddane zarówno logi systemowe jak i logi systemu IDS. Po wykonaniu wszystkich czynności wchodzących w skład procedury sporządzany jest protokół z przeprowadzonego badania.

A. Rozpoznawanie typu i wersji systemu operacyjnego (*fingerprint*)

Test A1

Test przeprowadzany jest przy pomocy narzędzia *nmap*. Parametry programu określają skanowanie komputera o adresie IP 10.1.9.12, oraz detekcję rodzaju systemu operacyjnego (parametr –O). Dodatkowe parametry (-vv) określają zwiększony poziom szczegółowości wyświetlanych informacji. Na rys 3 przedstawiono przykład zainicjowania testu A1.

```
nmap -O -vv 10.1.9.12

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-05-03 14:48 CEST

Host server (10.1.9.12) appears to be up ... good.
```

Rys. 3. Przykład zainicjowania testu A1

Test A2

Na podstawie informacji z programu *nmap* skanujemy serwer programem *xprobe2*, kierując test na port 23 (usługa ssh). Program *xprobe2* na podstawie charakterystycznych informacji ze stosu TCP i pliku konfiguracyjnego określa typ oraz wersję systemu operacyjnego. Na rysunku rys. 4 przedstawiono początkowy fragment raportu programu xprobe2.

```
xprobe2 -r -d 4 -v -P -T 23 10.1.9.12

Xprobe2 v.0.2rc1 Copyright (c) 2002-2003 fygrave@tigerteam.net, ofir@sys-security.com, meder@areopag.net

[+] Target is 10.1.9.12
```

Rys. 4. Przykład zainicjowania testu A2

B. Skanowanie portów

Test B1 – metoda *TCP connect()*

Otwierane są połączenia do nasłuchujących portów serwera. W tym celu atakujący wysyła pakiet z ustawioną flagą SYN na określony adres IP. Otwarty port docelowy odpowie pakietem z ustawionymi flagami SYN i ACK, natomiast zamknięty flagami RST i ACK. Na rys. 5 przedstawiono przykład zainicjowania testu B1.

```
nmap -sT -v 10.1.9.12
```

Rys. 5. Przykład zainicjowania testu B1

Test B2 – metoda *UDP scan*

Atakujący wysyła pakiety UDP do zadanych portów komputera testowanego. Używany w celu wykorzystania luk w RPC, NFS, tftp i innych. W odpowiedzi otrzymamy komunikat *ICMP Port unreachable* lub *ICMP Destination unreachable* świadczący o porcie zamkniętym. Na rys. 6 przedstawiono przykład zainicjowania testu B2.

```
nmap -sU -v -p \ 7,19,37,53,67,68,69,137,138,161,162,514,520,33434 10.1.9.12
```

Rys. 6. Przykład zainicjowania testu B2

Test B3 – metoda *Ping_Sweep*

Atakujący wysyła pakiety *ICMP echo request* typu 8, w celu wykrycia aktywnych komputerów. W odpowiedzi otrzymuje pakiet *ICMP echo* typu 0, jeśli pod zadanym adresem IP funkcjonuje komputer. Oprócz domyślnego komunikatu typu 8, *nmap* może wysłać pakiet z ustawioną flagą ACK na port 80 (opcja –sP). Jeśli w odpowiedzi otrzyma pakiet z flagą RST oznacza to działanie serwera. Na rys. 7 przedstawiono przykład zainicjowania testu B3.

```
nmap -sP -v 10.1.9.12
```

Rys. 7. Przykład zainicjowania testu B3

Test B4 – metoda *FIN scan*

Atakujący wysyła pakiety z ustawionym znacznikiem FIN. Odpowiedź serwera z ustawioną flagą RST,ACK informuje, iż port jest zamknięty. Niektóre systemy (np. Windows) zwracają taką odpowiedź również z portu otwartego. Na rys. 8 przedstawiono przykład zainicjowania testu B4.

```
nmap -sF -v server
```

Rys. 8. Przykład zainicjowania testu B4

Test B5 – metoda *XMAS*

Atakujący wysyła pakiety z ustawionym znacznikiem FIN,PUSH,URG. Odpowiedź serwera z ustawionymi flagami RST,ACK informuje o zamkniętych portach. Niektóre systemy (np. Windows) zwracają taką odpowiedź również z portu otwartego. Na rys. 9 przedstawiono przykład zainicjowania testu B5.

```
nmap -sX -v server
```

Rys. 9. Przykład zainicjowania testu B5

Test B6 – metoda *NULL*

Atakujący wysyła pakiety z ustawionym znacznikiem NULL. Odpowiedź serwera z ustawionymi flagami RST,ACK informuje o zamkniętych portach. Niektóre systemy (np. Windows) zwracają taką odpowiedź również z portu otwartego. Na rys. 10 przedstawiono przykład zainicjowania testu B6.

```
nmap -sN -v server
```

Rys. 10. Przykład zainicjowania testu B6

Test B7 – metoda *RPC scan*, *TCP reverse ident scan*

Atakujący wysyła polecenia NULL protokołu SunRPC do wszystkich otwartych portów atakowanego serwera (*RPC scan*) w celu zidentyfikowania serwisów RPC. Wykorzystuje demona *identd* celu do zidentyfikowania użytkownika będącego właścicielem aktywnych usług TCP. Na rys. 11 przedstawiono przykład zainicjowania testu B7.

```
nmap -v -sR -I server
```

Rys. 11. Przykład zainicjowania testu B7

C. Modyfikacja pakietów

Test C1

Program hping2 pozwala na wysłanie pakietu TCP/IP z dowolnie ustawionymi flagami. Parametr –M określa numer sekwencji, -S ustawioną flagę SYN, opcja -a pozwala na zmianę źródłowego adresu IP (*spoofing*). Na rys. 12 przedstawiono przykład zainicjowania testu C1.

```
hping2 10.1.9.12 -M 1111 -S -a 10.1.9.13
```

Rys. 12. Przykład zainicjowania testu C1

Test C2

Kolejne narzędzie pozwalające modyfikować pakiety TCP. Parametry polecenia są następujące: -D adres IP komputera docelowego, -S adres IP komputera źródłowego, -fS oznacza ustawienie flag SYN, -y pakiet kierowany jest do portu 22 (usługa ssh). Na rys. 13 przedstawiono przykład zainicjowania testu C2.

```
nemesis-tcp -v -S 10.1.9.13 -D 10.1.9.12 -fS -y 22
```

Rys. 13. Przykład zainicjowania testu C2

Test C3

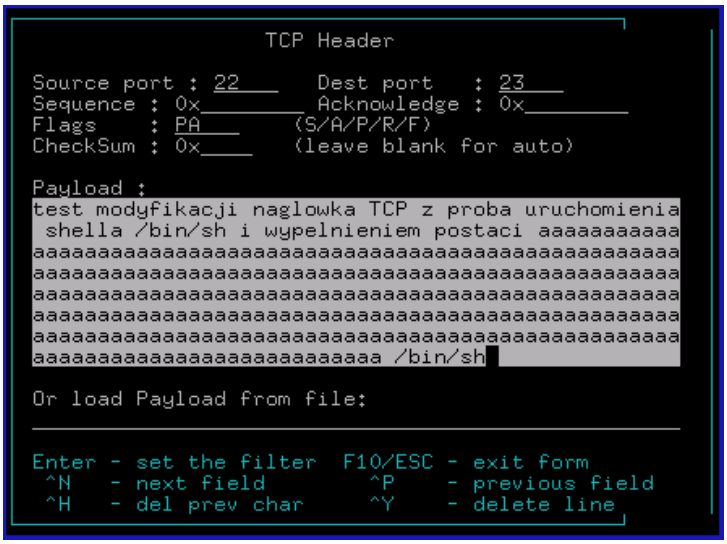
Kolejny test polega na spreparowaniu pakietu TCP z ustawionymi flagami PSH,SYN w nagłówku TCP, który kierowany jest na port 80 serwera 10.1.9.12. Adres źródłowy IP został zmodyfikowany i ustawiony na 10.1.9.13. Dodatkowo ustawiono numer sekwencji połączenia na 121212. Test ten jest wykonywany 10 razy. Na rys. 14 przedstawiono przykład zainicjowania testu C3.

```
num=1
while [ $num -le 10 ] ;do
nemesis-tcp -v -S 10.1.9.13 -D 10.1.9.12 -fP -fS -y 80  -s 121212
num=`expr $num + 1`
done
```

Rys. 14. Skrypt inicjujący test C3

Test C4

Modyfikacja nagłówka TCP programem ettercap. Na rysunku rys. 15 przedstawiono przykładowy pakiet utworzony za pomocą programu *ettercap*:



Rys. 15. Przykład modyfikacji pakietu narzędziem *ettercap* dla potrzeb testu C4

Test C5

Fragmentacja pakietu z wykorzystaniem narzędzia *fragtest*. Preparowany jest pofragmentowany pakiet o przesunięciu (offset) 8, 16, 24 bajty. Na rys. 16 przedstawiono przykład zainicjowania testu C5.

```
fragtest frag frag-new frag-old 10.1.9.12
```

Rys. 16. Przykład zainicjowania testu C5

Test C6

Kolejny test fragmentacji pakietu z wykorzystaniem skompilowanego programu *fragment_UDP*. Przeprowadzony jest atak typu *teardrop*. Jako parametry podawany jest adres IP serwera źródłowego, adres IP serwera docelowego oraz ilość generowanych pakietów. Na rys. 17 przedstawiono przykład zainicjowania testu C6.

```
fragment_UDP 1.1.1.1 10.1.9.12 –n 5
```

Rys. 17. Przykład zainicjowania testu C6

D. Fałszowanie ARP

Test D1

Test polega na sfałszowaniu tablicy ARP komputera atakowanego. Umożliwia to przekierowanie pakietów i ich ewentualną modyfikację przed dostarczeniem do celu. Na rys. 18 przedstawiono przykład zainicjowania testu C6.

```
arpspoof -t 10.1.254.254 10.1.9.12
arpspoof -t 10.1.9.12 10.1.9.10
```

Rys. 18. Przykład zainicjowania testu D1

E. Ataki DoS

Test E1

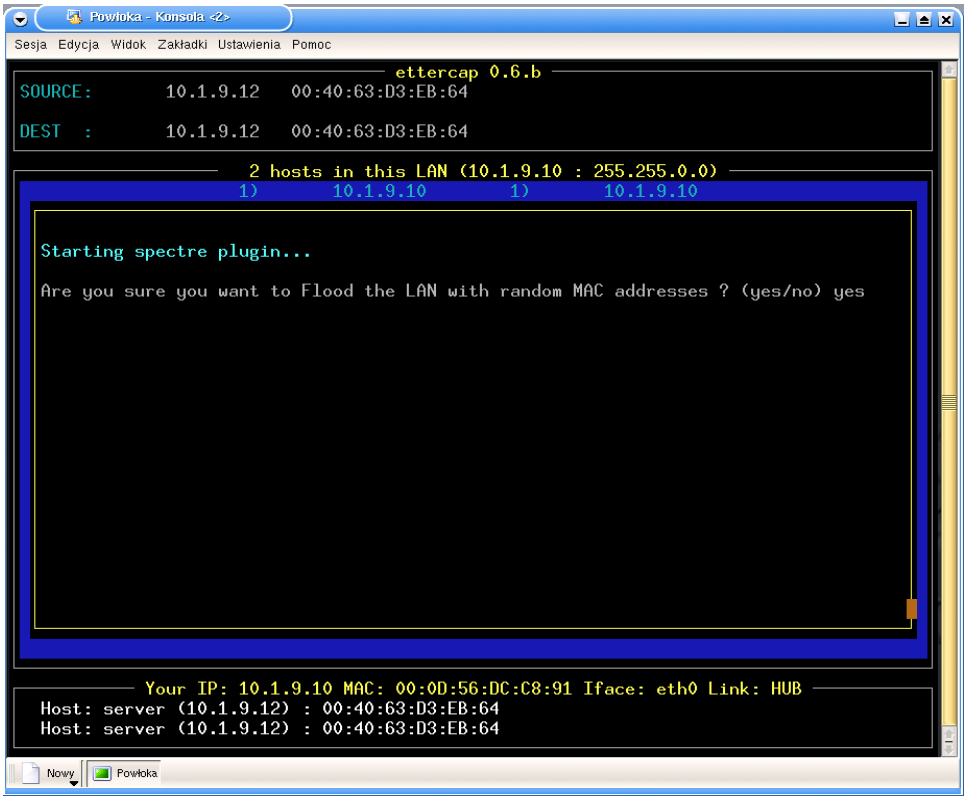
Do przeprowadzenia ataku wykorzystano program *ettercap* wraz z wbudowaną wtyczką (*plugin*) do przeprowadzenia ataku MAC flood. Na rys. 20 przedstawiono obraz okna programu ettercap w czasie przeprowadzania testu E1.

Test E2

Symulacja ataku typu *smurf*. Używamy programu *smurf* w wersji 4.0. Jako parametry programu należy podać adres IP serwera atakowanego, plik zawierający informację o adresie sieci (*broadcast*), ilość generowanych pakietów, opóźnienie pomiędzy kolejnymi pakietami w milisekundach, rozmiar pakietu. Na rys. 19 przedstawiono przykład zainicjowania testu E2.

```
smurf 10.1.9.12 bcast 20 2 512
```

Rys. 19. Przykład zainicjowania testu E2



Rys. 20. Przykład zainicjowania wykonania ataku *MAC flood* w teście E1

F. Atak na usługi

Test F1

Test polega na próbie odczytania pliku test.php z katalogu ROOT DIRECTORY serwera *apache*. Na rys. 21 przedstawiono przykład zainicjowania testu F1.

```
telnet 10.1.9.12 80

GET /test.php
```

Rys. 21. Przykład zainicjowania testu F1

Test F2

Test polega na próbie wykonania niedozwolonego zapytania sql do działającej bazy *MySql*. Na rys. 22 przedstawiono przykład zainicjowania testu F2.

```
telnet server 80
GET /przyklad.php?ID=5%20UNION%20select%20*%20from%20users;
```

Rys. 22. Przykład zainicjowania testu F2

Test F3

Przy pomocy dowolnej przeglądarki www wykonywana jest próba uruchomienia określonej funkcji interpretera *java-script*. Na rys. 23 przedstawiono przykład zainicjowania testu F3.

```
http://server/exploit.php?compress=%3Cscript%3Ealert('A%20L%20E%20R%20T')%3C/script%3E
```

Rys. 23. Przykład zainicjowania testu F3

Test F4

Atak z wykorzystaniem programu *apache_exploit*. Wykonuje on test przepełnienia bufora serwera *apache*, doprowadzając do niestabilności działania usługi. Na rys. 24 przedstawiono przykład zainicjowania testu F4.

```
apache_exploit 0x8f000 10.1.9.12
```

Rys. 24. Przykład zainicjowania testu F4

Test F5

Kolejny test wykonywany jest w stosunku do serwera usługi *samba*. Do tego celu używamy programu *samba_exploit* z parametrem określającym adres IP serwera atakowanego. Na rys. 25 przedstawiono przykład zainicjowania testu F5.

```
samba_exploit –h 10.1.9.12
```

Rys. 25. Przykład zainicjowania testu F5

Test F6

Atak przeprowadzany jest na serwer *wuftpd* (usługa ftp). Program *wuftpd_exploit* po próbie połączenia z atakowanym serwerem wykonuje polecenie MKD. Na rys. 26 przedstawiono przykład zainicjowania testu F6.

```
wuftpd_exploit –h 10.1.9.12 –u ftp –p ftp –n 21 -b
```

Rys. 26. Przykład zainicjowania testu F6

Test F7

Szereg ataków generowanych przez narzędzie *nikto* napisane w perlu. Skrypt wykonuje zapytania do serwera www, usiłując wylistować zawartość ROOT DIRECTORY serwera www, wykonać domyślne skrypty cgi, pobrać i wysłać dane używając metod GET oraz POST. Narzędzie ma zaimplementowane metody omijania systemów IDS. Na rys. 27 przedstawiono przykład zainicjowania testu F7.

```
nikto.pl –Cgidirs all –cookies –generic –host 10.1.9.12
```

Rys. 27. Przykład zainicjowania testu F7

Test F8

Atak z mechanizmem omijania systemów IDS. Włączona opcja „*Random URI encoding* (non-UTF8)". Na rys. 28 przedstawiono przykład zainicjowania testu F8.

```
nikto.pl –evasion 1 –host 10.1.9.12
```

Rys. 28. Przykład zainicjowania testu F8

Test F9

Atak z mechanizmem omijania systemów IDS. Włączona opcja „*Fake parameter*". Na rys. 29 przedstawiono przykład zainicjowania testu F8.

```
nikto.pl –evasion 5 –host 10.1.9.12
```

Rys. 29. Przykład zainicjowania testu F9

G. Inne testy

W ramach testów wykorzystanoy program *IDSwakeup*. Wykonuje on szereg ataków na usługi www, ftp. Badaniu poddane zostają także inne znane podatności systemów Unix. Narzędzie jest specjalnie dedykowane do testowania systemów wykrywania włamań.

4.4. Kryteria oceny badanych systemów wykrywania włamań.

Porównując badane systemy, przy konstruowaniu wniosków, pod uwagę należy wziąć następujące miary określające jakość badanego systemu:

- łatwość instalacji i konfiguracji. System oparty na otwartej architekturze powinien posiadać możliwość instalacji na wielu dystrybucjach systemów Unix/Linux,

- sposób zarządzania i administrowania modułami zbierającymi informację (sensor), modułem analizy i reakcji,
- sposób prezentacji wykrytych zdarzeń i informacji przydatnej w fazie późniejszej analizy potencjalnego ataku,
- możliwości elastycznej konfiguracji np. w przypadku zmiany adresacji IP w badanym środowisku, dodanie nowego sensora, tworzenie własnych reguł, itp.
- ilość fałszywych alarmów,
- ilość poprawnie wykrywanych incydentów.

Najważniejszą miarą, którą można określić liczbowo jest ostatnia z wymienionych. W tym celu przeprowadzone zostały opisane wyżej ataki, generujące zdarzenia, które powinny zostać wykryte przez badane systemy IDS.

5. Wyniki testów

Podsumowując kolejno każdy test bierzemy pod uwagę kryteria opisane poprzednio. Część ocen jest typowo subiektywna i zależna od osoby oceniającej. Poniżej przedstawiamy podsumowanie każdego testu dla obu badanych systemów IDS.

Łatwość instalacji

Instalacja systemu Snort nie sprawia żadnych kłopotów. Jest bardzo szczegółowo opisana i nawet dla początkującego administratora czy użytkownika, chcącego skorzystać z tego systemu, nie powinna być problemem. Poszczególne wymagane moduły systemu, tzn. baza danych, serwer *apache*, dodatkowe narzędzia i biblioteki oraz sam Snort wraz z przykładowym zbiorem reguł, są dostępne na zasadach „*open source*" w sieci. Pozwala to administratorom na pewne modyfikacje kodów źródłowych. Należy pamiętać, iż część parametrów konfiguracji standardowej (*default configuration*) może okazać nieoptymalna w konkretnym środowisku.

Podobnie przedstawia się sytuacja z instalacją programu Prelude IDS. Do badań została wykorzystana już zbudowana (skompilowana) dystrybucja tego systemu. Instalacja polegała wobec tego na wykonaniu polecenia systemu operacyjnego, za pomocą którego wszystkie niezbędne składniki systemu zostały pobrane, a następnie zainstalowane. Konfiguracja sprowadza się do modyfikacji określonych plików odpowiedzialnych za poprawne ustawienie i dostosowanie środowiska systemu operacyjnego i sieciowego.

Format raportów

Snort posiada możliwość wysłania określonego zapytania SQL do bazy przechowującej zapisane informacje. W sposób czytelny przedstawia informację o zarejestrowanym zdarzeniu. Pozwala na bardzo dokładną wizualizację atrybutów każdego pakietu odebranego z sieci. Konsola Snorta posiada kilka dodatkowych funkcji ułatwiających administratorowi analizę raportów. Jest to m.in. funkcja sortowania komunikatów alarmowych, wg dowolnego z dostępnych atrybutów i w dowolnym porządku. Możliwe jest również grupowanie określonych komunikatów według wybranych cech charakterystycznych, Dostępna jest również funkcja generowania różnego rodzaju wykresów.

Prelude IDS także został wyposażony w interfejs graficzny wykorzystujący przeglądarkę www. Reprezentacja informacji jest zwięzła i czytelna. Zawartość nagłówków pakietów może zostać zapisana w formacie XML. Prelude IDS posiada niewielką ilość wbudowanych filtrów, ale umożliwia tworzenie nowych. Raport jaki może zostać wygenerowany jest czytelny ale skromny. Tylko 10 najczęściej występujących typów ataków jest prezentowanych na wykresie kołowym. Dodatkowo istnieje możliwość tworzenia pewnych statystyk odnośnie stwierdzonych zagrożeń. Podobnie jak Snort, umożliwia grupowanie według różnych kryteriów.

Elastyczność konfiguracji.

Ewentualna szybka zmiana konfiguracji badanych systemów IDS nie wymaga wielu pracochłonnych i skomplikowanych czynności. Nie jest potrzebna powtórna kompilacja czy instalacja pakietów. Zmiany wprowadza się w jasno określonych plikach konfiguracyjnych w formacie tekstowym.

Skuteczność

Szczegółowe wyniki przeprowadzonych testów można znaleźć w [9]. W niniejszym opracowaniu zamieszczamy jedynie ich krótkie omówienie.

Test A1

Podczas testu przeskanowanych zostało 1659 portów. Program *nmap* wykrył 16 otwartych portów. Dokonał także próby określenia typu systemu operacyjnego oraz algorytmu do generowania kolejnych numerów sekwencji połączenia TCP/IP. Snort zarejestrował 21 alertów w 5 kategoriach. Prelude IDS zarejestrował 10 alertów w 8 kategoriach.

Na podstawie analizy logu z *tcpdump* można stwierdzić, że Snort poprawnie identyfikuje próby skanowania portów. Ilość wykrytych przez niego

alertów jest równa liczbie otwartych portów. Wykrył również dodatkowe działania identyfikujące typ systemu operacyjnego. Ponadto Snort zarejestrował dokładną liczbę prób nawiązania połączenia na otwarte porty i określił typ ataku jako „*EVASIVE RST detection*".

Natomiast Prelude IDS nie w pełni poprawnie identyfikuje wymuszane podczas przeprowadzania testu zdarzenia. Analiza logów wykazała, iż nie zarejestrował skanowania portu 705. Brak jest także tak dokładnej informacji o próbach nawiązania połączenia na otwartych portach.

Test A2

Zarówno Snort jak i Prelude rejestrują po jednym zdarzeniu, ale tylko Snort identyfikuje je poprawnie. Analizując sekwencję wymiany pakietów przedstawioną na rys. 30 wyraźnie widać dwukrotną próbę nawiązania połączenia. Po wysłaniu przez serwer pakietów z ustawionymi flagami SYN i ACK, komputer o adresie 10.1.9.10 wysyła pakiet z flagą RST. Pozostałe pakiety typu ICMP oraz ARP są standardowe i występują przy normalnej komunikacji w sieci.

```
10.1.9.10 -> Broadcast   ARP Who has 10.1.9.12?  Tell 10.1.9.10
10.1.9.12 -> 10.1.9.10   ARP 10.1.9.12 is at 00:40:63:d3:eb:64
10.1.9.10 -> 10.1.9.12   ICMP Echo (ping) request
10.1.9.12 -> 10.1.9.10   ICMP Echo (ping) reply
10.1.9.10 -> 10.1.9.12   TCP 19410 > telnet [SYN] Seq=0 Ack=0 Win=5840 Len=0
10.1.9.12 -> 10.1.9.10    TCP telnet > 19410 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
MSS=1460
10.1.9.10 -> 10.1.9.12   TCP [TCP ZeroWindow] 19410 > telnet [RST] Seq=1 Ack=17683591
Win=0 Len=0
10.1.9.10 -> 10.1.9.12   ICMP Echo (ping) request
10.1.9.12 -> 10.1.9.10   ICMP Echo (ping) reply
10.1.9.10 -> 10.1.9.12   ICMP Timestamp request
10.1.9.12 -> 10.1.9.10   ICMP Timestamp reply
10.1.9.10 -> 10.1.9.12   ICMP Address mask request
10.1.9.10 -> 10.1.9.12   ICMP Information request
10.1.9.12 -> 10.1.9.10   ICMP Destination unreachable
10.1.9.10 -> 10.1.9.12    TCP 14665 > telnet [SYN] Seq=0 Ack=0 Win=5840 Len=0
MSS=1460 TSV=573978 TSER=0 WS=0
10.1.9.12 -> 10.1.9.10    TCP telnet > 14665 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
MSS=1460 TSV=580321 TSER=573978 WS=0
10.1.9.10 -> 10.1.9.12   TCP [TCP ZeroWindow] 14665 > telnet [RST] Seq=1 Ack=22598765
Win=0 Len=0
10.1.9.12 -> 10.1.9.10   ARP Who has 10.1.9.10?  Tell 10.1.9.12
10.1.9.10 -> 10.1.9.12   ARP 10.1.9.10 is at 00:0d:56:dc:c8:91
```

Rys. 30. Zapis wymiany pakietów podczas realizacji testu A2

Test B1

Zarówno Snort jak i Prelude IDS poprawnie zidentyfikowały test polegający na skanowaniu określonych portów. Drobna różnica w ilości i pisie występujących zdarzeń może wynikać ze sposobu definiowania bazy sygnatur
w obu badanych systemach.

Test B2

Kolejny test polegający na skanowaniu 14 portów UDP pozwala zaobserwować dokładniejszą analizę po stronie Prelude IDS. Snort rejestruje 3 alerty wskazujące na próby przesłania na określone porty oraz próbę ataku DoS. Natomiast Prelude dokładnie rejestruje próbę skanowania określając przy tym ilość portów oraz charakter ataku. Liczba unikalnych alertów jest taka sama jak w przypadku Snorta.

Test B3

Snort oraz Prelude IDS rejestrują poprawnie takie samo zdarzenie – ICMP PING NMAP.

Test B4

Skanowanie z ustawioną flagą FIN jest poprawnie identyfikowane przez oba systemy wykrywania włamań. Różnica w ilości zarejestrowanych zdarzeń może wynikać z innej konfiguracji preprocesora odpowiedzialnego za wykrycie skanowania.

Test B5

Podczas tego badania sensory Snorta i Prelude identyfikują poprawnie występujące zdarzenia. Podobnie jak w poprzednim teście ilość alertów w przypadku Prelude IDS jest prawie 4-krotnie większa niż w przypadku Snorta.

Test B6

Identyfikacja zdarzeń przez oba systemy IDS jest poprawna. Tak jak we wcześniejszych testach ilość alertów zarejestrowanych przez Prelude (1527) jest 4 razy większa niż alertów zarejestrowanych przez Snorta (427).

Test B7

Liczba unikalnych zdarzeń zarejestrowanych podczas testu Snorta wynosi 7, natomiast Prelude IDS rejestruje 4 unikalne alerty. Snort poza poprawnym zarejestrowaniem typu ataku wskazuje 17 skanowanych portów. Ta część skanowania została przez Snorta zarejestrowana jako „*EVASIVE RST detection*”. Prelude informuje o przeprowadzonym skanowaniu.

Test C1

Ilość alertów generowanych przez badane systemy wykrywania włamań zależy od czasu trwania testu. W związku z tym pod uwagę brana jest poprawność identyfikacji zdarzenia, bez rozpatrywania ilości zarejestrowanych zdarzeń. Oba systemy poprawnie zidentyfikowały zachodzące zdarzenia. Dodatkowo Prelude IDS zarejestrował alert związany z funkcjonowaniem modułu *ArpSpoof* odpowiedzialnego za analizę i porównanie adresów IP oraz MAC.

Test C2

Snort rejestruje jedno zdarzenie typu „*EVASIVE RST detection*". Jest to poprawna identyfikacja zaistniałej sytuacji. Natomiast Prelude IDS rejestruje alert odmiennego typu – „*Directed ARP request*". Wynika to z próby nawiązania połączenia pomiędzy komputerami o adresie IP 10.1.9.13 i 10.1.9.12, podczas gdy atak prowadzony jest z komputera 10.1.9.10. Jest to także poprawna identyfikacja zdarzenia..

Test C3

W tym teście Snort także poprawnie rejestrował ilość i typ zdarzeń. Natomiast Prelude IDS, podobnie jak w teście 3.2 rejestruje odmienny, lecz również poprawny typ zdarzenia. W tym miejscu można już stwierdzić, iż jakakolwiek modyfikacja adresu źródłowego będzie przez sensor Prelude identyfikowana jako „*Directed ARP request*".

Test C4

Snort poprawnie wykrywa w przesyłanym pakiecie ciąg „/bin/sh". Natomiast Prelude IDS zauważył modyfikację adresu źródłowego oraz wysłanie pakietów ICMP typ 8. Jeśli na etapie analizy ramki ethernet możliwe jest zablokowanie takiego pakietu, to jest to skuteczniejsza metoda obrony przed tego typu atakami niż realizowana przez Snorta. Za niedogodność można poczytać skąpą ilość informacji dostarczaną przez sensory Prelude.

Test C5

Test polegający na fragmentacji pakietu został poprawnie zidentyfikowany przez Snorta. Ilość unikalnych alertów wygenerowanych przez sensor wynosi 2. Prelude IDS wykrył także 2 zdarzenia, w tym atak typu „*IP Fregmantation attack*". Niestety alert typu „*Directed ARP request*" nie wynika z modyfikacji adresu źródłowego. Jest to fałszywy alarm wygenerowany przez sensor Prelude.

Test C6

Zarówno Snort jak i Prelude IDS poprawnie identyfikują typ przeprowadzonego ataku jako „*Teardrop attack*".

Test D1

Podczas wykonywania badania uruchomione zostało polecenie *ping*, aby lepiej zaobserwować wynik ataku. Zarówno Snort jak i Prelude IDS poprawnie rejestrują i identyfikują zachodzące zdarzenia.

Test E1

Także w tym badaniu sensory Snorta i Prelude IDS poprawnie rejestrują typ prowadzonych ataków. Liczba wygenerowanych alertów uzależniona jest od czasu trwania testu.

Test E2

Snort poprawnie wykrywa incydenty związanych z przeprowadzanym badaniem. Prelude IDS rejestruje jedno zdarzenie związane z modyfikacją adresu źródłowego. Jest to również poprawna identyfikacja.

Test F1

Test polegający na próbie odczytania domyślnego pliku test.php. Zarówno Snort jak i Prelude IDS poprawnie identyfikują to zdarzenie.

Test F2

Kolejny atak z wykorzystaniem przeglądarki www polega na wykonaniu niedozwolonego zapytania do bazy danych MySQL. Snort poprawnie identyfikuje i rejestruje zdarzenie. Natomiast Prelude IDS nie wykrywa żadnego incydentu. Dla pewności wykonano test kilkakrotnie. Powodem takiego stanu może być nieaktualna baza sygnatur lub brak opisu w postaci sygnatury dla tego ataku.

Test F3

Oba systemy wykrywania włamań poprawnie rejestrują incydent, polegający na przekazaniu w zapytaniu URL funkcji *javascript*.

Test F4

Ilość alertów zarejestrowanych przez Snort i Prelude jest różna. Snort wykrył w 3 grupach 139 zdarzeń. Natomiast Prelude IDS w 5 grupach wykrył 537 zdarzeń. Różnica wynika z budowy modułu analizy sygnatur obu systemów wykrywania włamań, oraz z możliwości zaliczenia przez Snorta pojedynczej sesji nawiązania połączenia jako pojedynczego incydentu.

Test F5

Test polega na dwukrotnej próbie uzyskania dostępu do zasobów udostępnionych przez serwer „samba". Snort zdołał zarejestrować 4 alerty w 2 kategoriach, tym samym poprawnie identyfikując zdarzenia. Natomiast Prelude IDS nie wykrył żadnych zagrożeń. Przyczyną jest brak odpowiednich sygnatur dla tego typu ataków.

Test F6

Snort podczas badania zarejestrował 69 alertów w 5 kategoriach. Identyfikacja zdarzeń jest prawidłowa. Prelude IDS wykrył w czasie testu 110 zdarzeń w 4 grupach. W tym przypadku identyfikacja zdarzeń także jest prawidłowa, jednak liczba wszystkich zarejestrowanych alertów jest znacznie większa niż w przypadku Snorta. Powodem może być tendencja do wielokrotnego zliczania alertów w ramach jednej nawiązanej sesji TCP.

Test F7

Test ten posiada znamiona testu obciążenia serwera www wraz z mechanizmami testowania potencjalnych luk w systemie. Snort rejestruje 936 alertów w tym 178 unikalnych. Natomiast Prelude IDS - 327 alerty, w tym 80 unikalnych. Trudno określić przyczynę takiego wyniku, ale może nią być sposób przetwarzania kolejnych pakietów przez sensor Prelude.

Test F8

Podobny test do poprzedniego z opcją „*Random URI encoding*". Obserwuje się różnicę ilości zarejestrowanych w obu systemach IDS zdarzeń. Snort wykrył 92 unikalne zdarzenia, natomiast Prelude IDS 68.

Test F9

Jeszcze jeden test z wykorzystaniem tego samego narzędzia, lecz z włączonym parametrem omijania systemów IDS. Snort zarejestrował 183 alerty w 26 unikalnych grupach. Natomiast Prelude zarejestrował 777 alertów w 38 grupach. Jeszcze raz potwierdza się teza o tendencji Prelude IDS do wielokrotnego zliczania incydentów w ramach tego samego nawiązanego połączenia. Jest to objaw niepożądany.

Test G

Ostatni z testów wykorzystuje specjalne narzędzie do badania systemów IDS. Snort zarejestrował podczas wykonywania testu 10361 alertów w 40 unikalnych grupach. Natomiast Prelude IDS zarejestrował nieco mniej, bo 6814 alertów w 19 grupach.

6. Podsumowanie

Dobór odpowiedniego narzędzia do zabezpieczenia sieci jest bardzo trudny i dla administratora sieci stanowi pewne wyzwanie. Jako przykład staraliśmy się dobrać badane systemy wykrywania włamań, aby możliwie najlepiej chroniły zasoby sieciowe, posiadały podobne funkcje i możliwości, a przy tym były elastyczne i wysoce konfigurowalne. Wybór padł na najbardziej popularne wśród oprogramowania „*open source*" systemy IDS przeznaczone na platformę Unix/Linux.

Można zauważyć, że jakość wykonania instalacji i konfiguracji ma wielkie znaczenie dla poziomu bezpieczeństwa sieci. Dobór odpowiednich parametrów konfiguracji systemu IDS dla danego środowiska sieciowego musi odbywać się na zasadzie próbkowania kolejnych możliwości i analizy konfiguracji. Poprzez wykonanie symulacji ataków można dobrać odpowiednią, zgodną z polityką bezpieczeństwa, optymalną konfigurację systemu.

Aby wykonać badanie systemu należy określić procedury testowe, które zasymulują ataki przeprowadzane na środowisko sieciowe. Dobór odpowiednich testów, ich późniejsza interpretacja i opracowane w ten sposób metody zapobiegania atakom, pozwolą zaoszczędzić wielu kłopotów związanych z naruszeniem bezpieczeństwa i kompromitacją systemu informatycznego.

Wśród wybranych systemów wykrywania włamań nie ma żadnego systemu komercyjnego. Przy odpowiednim doborze konfiguracji, systemy typu „*open source*" mogą z nimi konkurować. Zaletą oprogramowania typu „*open source*" jest dostępność do źródeł, możliwość modyfikacji i pisania własnych funkcji zwiększających funkcjonalność. Wsparciem dodatkowym jest funkcjonowanie wielu grup dyskusyjnych zajmujących się podobną problematyką.

Systemy wykrywania włamań uzupełniają inne mechanizmy ochronne środowiska sieciowego. Stały się nieodłącznym dodatkiem do funkcjonujących już od dawna zapór ogniowych, z którymi bardzo często współpracują. Aby jednak taka konfiguracja była skuteczna, należy przeprowadzić szereg testów, w których symuluje się różne ataki, bada się sposób reakcji sensorów, poprawność interpretacji ataku i poszukuje się coraz to nowszych, lepszych metod obrony przed zagrożeniami występującymi w sieci. Przykładową metodykę badania systemów wykrywania włamań dla środowiska Unix/Linux staraliśmy się przedstawić w niniejszym opracowaniu.

Literatura

[1] Amoroso E.: *Wykrywanie intruzów*, RM, 1999.

[2] Liderman K.: *Bezpieczeństwo teleinformatyczne,* IAR WAT 2001.

[3] Ahmad D.R.M., I inni: *Hack Proofing Your Network*, Helion 2002.

[4] awcewicz M.: *Techniki skanowania sieci komputerowych*, Software 2.0 09/2001.

[5] Kruk D.: *Nowe techniki skanowania sieci* – Pckurier 8/2002.

[6] Magalhaes R.M.: *Host-Based IDS vs Network-Based IDS*, Windows Security 06/2003.

[7] Müller K.: *IDS - Intrusion Detection System, Part I* - Linux Focus 05/2003.

[8] Müller K.: *IDS - Intrusion Detection System, Part II* - Linux Focus 06/2003.

[9] Klepka A.: *Analiza porównawcza systemów wykrywania włamań dla środowiska Unix/Linux*, praca magisterska. Biblioteka Wojskowej Akademii Technicznej.

Recenzent: prof. dr hab. inż. Włodzimierz Kwiatkowski

Praca wpłynęła do redakcji 10.10.2004