МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Навчально-науковий інститут інноваційних освітніх технологій
Кафедра комп'ютерної інженерії

ДОВГИЙ Віталій Васильович

Алгоритми виявлення процедури сканування портів в корпоративній комп'ютерній мережі / Algorithms for scanning ports detecting procedure in the corporate computer network

спеціальність: 123 - Комп’ютерна інженерія
магістерська програма - Комп’ютерна інженерія

Магістерська робота

Виконав студент групи КІм-21
ДОВГИЙ В.В.
Науковий керівник: д.т.н. професор,
В.В. Яцків

Магістерську роботу допущено до захисту:

ТЕРНОПІЛЬ -2018

РЕЗЮМЕ

Магістерська робота на тему «Алгоритми виявлення процедури сканування портів в корпоративній комп'ютерній мережі» зі спеціальності 123 «Комп'ютерна інженерія» написана обсягом 94 сторінок і містить 27 ілюстрації, 8 таблиць, 3 додатки та 51 джерел за переліком посилань.

Метою роботи є дослідження алгоритмів процедури виявлення сканування портів в корпоративній комп'ютерній мережі, а також удосконалення вже існуючого.

Методи досліджень. Для розв'язання поставлених задач у магістерській роботі використано: алгоритми та утиліти сканування портів, методи сканування портів, мережеві протоколи, фільтрація портів інтернет - провайдерами, програмне забезпечення яке здійсює сканування а також виявлення процедури сканування портів, основи TCP/IP, принципи забезпечення інформаційної безпеки.

Результати дослідження: удосконалено алгоритм виявлення процедури сканування портів а також, детально описано структуру утиліти виявлення атак на мережу.

Результати роботи можуть бути використані при захисті як локальної чи корпоративної мережі підприємства так і домашнього комп'ютера, а також в навчальному процесі.

Орієнтовні напрямки розвитку досліджень: підвищення швидкодії реагування на атаки способом сканування портів; оптимізація апаратних затрат на реалізацію виявлення сканування.

КЛЮЧОВІ СЛОВА: АЛГОРИТМ, СКАНУВАННЯ ПОРТІВ, МЕРЕЖЕВІ АТАКИ, МЕРЕЖЕВІ ПРОТОКОЛИ.

RESUME

Master's work on the theme "Algorithms for detecting the procedure of port scanning in the corporate computer network" from the specialty 123 "Computer Engineering" is written in volume of 94 pages and contains 27 illustrations, 8 tables, 3 annexes and 51 sources according to the list of references.

The purpose of the work is to study the algorithms of the procedure for detecting port scans in the corporate computer network, as well as the improvement of existing ones.

Research methods. In order to solve the tasks set in the master's work, algorithms and utilities of port scanning, ports scanning methods, network protocols, Internet porting filtering, software that scans, and also reveals the procedure of port scanning, the basis of TCP / IP, principles of information provision security.

Results of the research: improved algorithm for detecting the procedure for scanning ports, as well as a detailed description of the structure of the utility for detecting attacks on the network.

The results of the work can be used to protect both the local or corporate network of the enterprise and the home computer, as well as in the learning process.

The approximate directions of research development: increasing the speed of response to attacks by way of port scanning; optimization of hardware costs for the implementation of detection of scanning.

KEY WORDS: ALGORITHM, PORT SCANNING, NETWORK ATTACKS, NETWORK PROTOCOLS.

ЗМІСТ

ВСТУП

В Інтернеті щодня з’являється нове шкідливе програмне забезпечення. У зв’язку з цим захист персональних комп’ютерів і цифрових активів стає важливою задачею. Першою фазою більшості комп’ютерних атак є розвідка. Одним із механізмів здійснення розвідки є сканування портів, яке дозволяє зловмисникові з’ясувати, які сервіси працюють в цільовій системі, а значить, підготувати і провести цілеспрямовану атаку проти виявлених сервісів і їх вразливостей. Отже, боротися з розвідкою, у вигляді сканування портів є актуальною задачею.

Більшість користувачів не звертають увагу на структуру Інтернету і його компонентів, а тільки використовують послуги, що надаються їм операційною системою або додатками. Проте, існує невелике число просунутих користувачів, які використовують свої знання для вивчення потенційних вразливостей системи. Програмісти можуть поставити під загрозу вразливі вузли мережі і можуть використовувати їх як свої, або використовувати їх в якості інструментів для майбутніх атак. Одним з популярних методів пошуку жертв є сканування портів. Сканування портів може бути визначено як ворожий Інтернет пошук відкритих дверей або портів, через які зловмисники отримують доступ до комп'ютерів. Цей метод полягає у відправці повідомлень на порт і прослуховування відповіді. Отримана відповідь вказує на стан порту і може використовуватися для визначення операційної системи та іншої інформації, що має відношення до майбутньої атаки.

Мета і завдання дослідження. Метою роботи є дослідження алгоритмів виявлення сканування портів в комп’ютерній локальній мережі.

Досягнення визначеної мети передбачає вирішення таких завдань:

– провести аналіз можливих атак за допомогою сканування портів;

– дослідити алгоритм сканування відкритих портів в комп’ютерній мережі;

– дослідити алгоритм виявлення сканування портів в комп’ютерній мережі;

– розробити унікальні правила утиліти для виявлення сканування відкритих портів підходящі для даної комп’ютерної мережі;

– реалізувати написані правила у тестуванні алгоритму.

Об’єкт дослідження– процеси сканування та виявлення сканування портів в комп’ютерній мережі.

Предмет дослідження – методи та алгоритми виявлення та блокування сканування портів в комп’ютерній мережі.

Методи досліджень. Для розв’язання поставлених задач у магістерській роботі використано: алгоритми та утиліти сканування портів, методи сканування портів, мережеві протоколи.

Наукова новизна одержаних результатів. Удосконалено алгоритми виявлення та блокування сканування відкритих портів за рахунок написання унікальних правил, що дозволило збільшити виявлення атак на мережу.

Практичне значення отриманих результатів. Розроблено унікальні правила та реалізовано в утиліті Snortc для забезпечення безпеки мережі у лабораторії.

Публікації та апробація ДР. Довгий В.В., Небесний І.В. Алгоритми сканування портів у корпоративній комп’ютерній мережі. Актуальні задачі сучасних технологій: семінар комп’ютерні науки та інформаційні технології. – Тернопіль: ТНЕУ, 2018. – С. 27 [1].

В першому розділі було досліджено всі типи мережевих атак які здійснюються за допомогою сканування портів у мережі, а також було розглянуто архітектуру роботи даних атак.

В другому розділі було досліджено алгоритми утиліт для виявлення таких атак як сканування портів в корпоративній мережі, а також архітектуру та особливості роботи трьох популярних утиліт таких як: PSAD, Snort, PortSentry.

В третьому розділі було обрано одну з утиліт яка найбільш підходить нам і більш детально досліджено алгоритм та особливість роботи утиліти Snort. Розроблено унікальні правила для корпоративної мережі і протещено роботу.

1. ТИПИ СКАНУВАННЯ ПОРТІВ В КОМП’ЮТЕРНІЙ ЛОКАЛЬНІЙ МЕРЕЖІ

У зв'язку з існуванням великої кількості різних протоколів і безліччю реалізацій кожного з них для різних платформ, запуск ефективної атаки часто починається з окремого процесу виявлення потенційних жертв.

Для виявлення факту сканування портів в інформаційній системі була розроблена математична модель для подальшого створення програмного комплексу, що використовує аналіз мережевого трафіку для виявлення сканування.

Розроблена математична модель використовує перехоплені пакети мережевого трафіку для аналізу на наявність аномалій і виявлення фактів сканування портів зловмисником.

З перехоплених пакетів виділяються IP-адреси і прапори і відбувається формування таблиць відповідності IP-адрес і комбінацій їх прапорів. Програмний засіб перехоплює весь трафік, проходить через мережевий пристрій. Необхідно запам’ятовувати дані пакетів, щоб побудувати еталонну модель поведінки мережі. Після навчання програми, отримані пакети аналізуються методом NT і заповнюються таблиці, що містять 2 поля-IP і комбінації прапорів.

Щоб виявити віддалене сканування портів і визначити тип сканування, отримані результати порівнюються за певним алгоритмом. Після цього,в разі виявлення аномалій, запускається алгоритм пошуку комбінацій в раніше заповнених таблицях. Модуль перехоплення трафіку підключається до мережевого інтерфейсу і веде перехоплення всі вхідні і вихідні пакети.

Модуль аналізу NT-методом робить аналіз перехоплених пакетів використовуючи математичну модель, описану вище. Модуль розрахунку еталонної моделі мережі вираховує індекс аномальності за формулою і зберігає його для подальшого порівняння з поточним індексом аномальності.

Модуль заповнення журналу заповнює таблиці на основі перехоплених пакетів.

Модуль виявлення зловмисника виявляє аномалії в трафіку, отримуючи поточний індекс аномальності і порівнюючи його з еталонним індексом, з наданими модулем розрахунку еталонної моделі мережі.

1.1 Види сканування портів

Сканування дозволяє здійснювати пошук каналів передачі даних. Ідея сканування полягає в тому, щоб дослідити якомога більше потенційних каналів зв'язку і визначити, які саме знаходяться в стані очікування з'єднання. Як канал зв'язку теоретично може виступати будь-яка сукупність приймально-передавального обладнання із середовищем передачі даних. Ми розглянемо лише можливу його реалізацію, що представляє собою комп'ютерний термінал (порт), підключений до мережі по комутованій або виділеної лінії.

Термін «порт» є абстрактним поняттям, використовуваним для спрощеного опису механізму встановлення з'єднання між комп'ютерами. Дотримуючись наведеної вище термінології, порт являє собою потенційний канал передачі даних. Використання механізму портів істотно полегшує процес встановлення з'єднання і обміну інформацією між комп'ютерами. Як і всюди, в організації механізму портів є свої недоліки. Будь-який користувач має можливість досліджувати мережеве оточення сервера методом опитування його портів. Достатньо лише надіслати «лавину» пакетів на всі можливі номери портів сервера (1-65535), і по тому, від яких портів будуть (або не будуть) отримані відповіді, визначити відкриті порти і служби, що працюють на досліджуваному сервері.

Існує велика кількість алгоритмів для пошуку активних портів хоста і відповідних їм служб. Розглянемо найбільш часто використовувані алгоритми сканування і розберемо їх переваги та недоліки.

Методи сканування TCP-портів. Визначення стану сервера методом ICMP-сканування. Перед безпосереднім скануванням портів віддаленого комп'ютера необхідно з'ясувати його стан - працює він у мережі чи ні. Особливо це важливо

при скануванні групи хостів або при скануванні певного сегмента мережі, де крім визначення функціонуючих хостів необхідно також визначити їх адреси.

Для цього необхідно відправити специфічний запит, що означає, що користувачеві необхідна інформація про стан сервера (або хоста) і справності мережевих засобів, що забезпечують маршрутизацію пакетів. У мережах, організованих на базі стека протоколів TCP / IP, для цієї мети використовується протокол ICMP. Даний протокол є допоміжним і дозволяє маршрутизатора повідомляти кінцевому вузлу про помилки або непередбачених ситуаціях, які мали місце при передачі IP-дейтаграми від цього вузла. Обмін інформацією між маршрутизатором і вузлом реалізований за допомогою так званого ICMP-повідомлень. Крім повідомлень про помилки, в протоколі ICMP передбачений ряд стандартних запитів, що дозволяють хосту отримати різного роду інформацію про стан об'єктів мережі.

Як згаданого вище запиту хост відправляє серверу ICMP-повідомлення і чекає отримання відповіді, також це ICMP-повідомлення (так званого ICMP-відлуння). Повідомлення ICMP користувач може сформувати програмним способом або використовувати для цього засоби операційної системи.

Для програмної реалізації даного методу користувачеві необхідно знати деякі особливості побудови подібних запитів. Будь-яке ICMP-повідомлення має два рівні інкапсуляції в IP-дейтаграмму (див. таблиця1.1).

Таблиця 1.1 – Інкапсуляція ICMP-повідомлення

<table>
<tr><td></td><td></td><td>ЗаголовокICMP-повідомлення</td><td>ДаніICMP-повідомлення</td></tr>
<tr><td></td><td>ЗаголовокIP-дейтаграми</td><td colspan="2">Область даних IP-дейтаграми</td></tr>
<tr><td>заголовок кадру канального рівня</td><td colspan="3">Область даних кадру</td></tr>
</table>

На початку будь-якого ICMP-повідомлення знаходяться три поля: «Тип повідомлення», «Код» (причина помилки) і «Контрольна сума». Поле "Тип"

визначає зміст ICMP-повідомлення і відповідний йому формат. Значення цього поля наведені в таблиці 1.2.

Таблиця 1.2 – Значення коду ICMP-повідомлень

код	Значення
0	Відповідь на відлуння
3	Одержувач недосяжний
4	Придушення джерела
5	Зміна маршруту
8	Запит луни
11	Перевищено час дейтаграми
12	Помилка параметрів дейтаграми
13	Запит тимчасової мітки
14	Відповідь на запит тимчасової мітки
17	Запит маски адреси
18	Відповідь на запит маски адреси

В даному методі використовуються типи повідомлень «Запит луни» (8) і «Відповідь на відлуння» (0). Зазвичай запит луни і пов'язаний з ним відповідь використовуються для перевірки досяжності одержувача (тобто скануючого сервера) IP-дейтаграми і його здатності відповідати на запити. Формат ICMP-повідомлень «запит луни» і «відповідь на запит луни» показаний в таблиці 1.3.

Таблиця 1.3 – Формат повідомлення "запит на відлуння" і відповіді на нього

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Тип (8 або 0)							Код (0)												Контрольна сума								
Ідентифікатор																				Послідовний номер							
Необов’язкові дані																											

Поле «Необов'язкові дані» має змінну довжину і містить дані, які необхідно повернути відправнику. Поля «Ідентифікатор» і «Послідовний номер» використовуються відправником для перевірки відповідності відповідей запитам.

Так як запит луни і відповідь на нього передаються в IP-дейтаграмах, то успішний прийом відповіді свідчить про працездатність основних частин транспортної системи, тобто виконана маршрутизація, працездатні всі проміжні маршрутизатори, одержувач активний і працює коректно, програмне забезпечення протоколів IP і ICMP виконує свої функції. Іншими словами, при отриманні відповіді від об'єкту сканування сервера на відправлений йому «запит луни» свідчить про те, що сервер працює і, можливо, очікує запит на з'єднання.

У багатьох операційних системах програма, яка використовується для посилки запиту луни, називається ping. З цієї причини запит луни називають ще ping-запитом. Програма ping спеціально призначена для визначення стану будь-якого об'єкта мережі, що має власний IP-адреса.

При скануванні великих мереж затримка в часі між посилкою запиту й одержанням відповіді на нього може перевищити встановлене значення, в результаті чого ping може неправильно інтерпретувати стан об'єкта і вказати на відсутність до нього доступу. Аналогічна ситуація може статися при «ручному» формуванні запиту луни. В обох випадках цього можна уникнути, збільшивши час очікування відповіді на ping-запит.

Сканування TCP-портів функцією connect ().Даний метод використовувався в самому початку розвитку технології сканування, проте до цих пір є основним і єдиним в деяких операційних системах (Windows), що підтримують механізм сокетів, для сканування портів по протоколу TCP. Функція connect () дозволяє хосту з'єднатися з будь-яким портом сервера. Якщо порт, зазначений в якості параметра функції, прослуховується сервером (тобто порт відкритий для з'єднання), то в результаті виконання функції connect (n) буде встановлено з'єднання з сервером через порт, n. В іншому випадку, якщо з'єднання не встановлено, то порт з номером n є закритим.

Цей метод має деякі переваги. По-перше, його може застосувати будь-який користувач, що не володіє ніякими привілеями на хості. По-друге, даний метод забезпечує досить високу швидкість дослідження. Послідовний перебір портів шляхом виклику функції connect () для кожного номера порту, визначення його стану і закриття з'єднання - досить довгий процес. Однак його можна прискорити,

застосувавши метод «паралельного перегляду» з використанням не блокування введення / виведення (non-blocked I / O). Такий метод дозволяє практично одночасно визначити стан всіх портів сервера.

Великим недоліком даного методу є можливість виявлення і фільтрації такого роду сканування, причому зробити це досить легко. Log-файл від сканованого сервера вкаже службам, відповідальним за зовнішні підключення, на наявність численних запитів на з'єднання з одного і того ж адреси і помилок створення з'єднання з ним, оскільки хост досліджує після створення з'єднання з сервером відразу ж обриває його. Служби зовнішніх підключень, в свою чергу, негайно заблокують доступ до сервера для хоста з даними адресою.

Сканування TCP-портів прапором SYN**.** Даний метод відомий ще як «сканування з встановленням наполовину відкритого з'єднання» (half-open scanning), оскільки повне встановлення TCP-з'єднання не проводиться. Розглянемо схему створення TCP-з'єднання, описану в протоколі TCP. У початковому стані сервер «прослуховує» порти в очікуванні з'єднання. З'єднання між хостом і сервером не встановлено.

Перший етап (рисунок 1.1): хост посилає серверу SYN-пакет із зазначенням власного номера черги.

Рисунок 1.1 – Перший етап встановлення з'єднання

Другий етап (рисунок 1.2): сервер, прийнявши запит на з'єднання, посилає хосту підтвердження і дані для синхронізації зі свого боку.

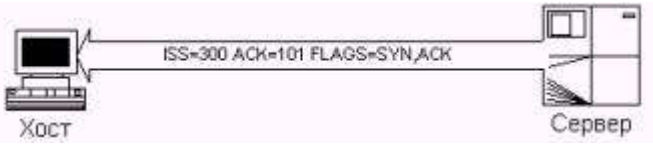

Рисунок 1.2 – Другий етап встановлення з'єднання

Третій етап (рисунок 1.3): хост, прийнявши пакет синхронізації від сервера, посилає йому підтвердження про прийом.

Рисунок 1.3 – Третій етап встановлення з'єднання

Процес, розглянутий вище, називається триступеневої синхронізацією (3-way handshaking), і служить для встановлення з'єднання по протоколу TCP між двома будь-якими об'єктами мережі Інтернет. Після цього хост передає серверу дані (рисунок 1.4).

Рисунок 1.4 – Хост передає дані серверу

Як видно, процес встановлення з'єднання передбачає взаємний обмін пакетами синхронізації між сервером і хостом. Кожна зі сторін повинна отримати початковий номер черги (ISS) «партнера» і надіслати підтвердження. Пакет синхронізації є сформоване за правилами протоколу TCP повідомлення з виставленим в ньому прапором SYN (або SYN і ACK для підтвердження синхронізації).

Алгоритм сканування наступний. Хост відправляє на певний порт сервера SYN-пакет, як би маючи намір створити з'єднання, і очікує відповідь. Наявність у відповіді прапорів SYN | ACK означає, що порт відкритий і прослуховується сервером. Отримання у відповідь TCP-пакета з прапором RST означає, що порт закритий і не прослуховується.

У разі прийому SYN | ACK-пакета хост негайно відправляє RST-пакет для скидання встановлюється сервером з'єднання і не продовжує процес обміну

пакетами. Таким чином, проводиться перевірка здатності сканування сервера встановити з'єднання через порт.

Перевага даного методу полягає в тому, що лише деякі сервери здатні зареєструвати такого роду сканування без використання спеціальних засобів захисту. Метод можливо використовувати тільки в разі, якщо на хості, з якого виробляється сканування, встановлена операційна система з сімейства UNIX. Крім того, користувач повинен мати статус root, в іншому випадку користувач просто не зможе програмно сформувати одиночний SYN-пакет.

Сканування TCP-портів прапором FIN. Як вже говорилося, лише деякі сервери здатні відстежити спробу SYN-сканування їх портів. Так, деякі файрволли і пакетні фільтри «очікують» підроблені SYN-пакети на закриті порти захищеного ними сервера, і спеціальне програмне забезпечення типу synlogger або courtney розпізнає спробу SYN-сканування. Якщо сервер обриває з'єднання після опитування декількох портів, використовується FIN-сканування.

У цьому методі використовуються FIN-пакети, які використовуються в процедурі закриття з'єднання. Пакет передбачає установку в TCP-повідомленні прапора FIN. Розглянемо процедуру закриття з'єднання.

У початковому стані хост передає серверу дані (рисунок 1.5).

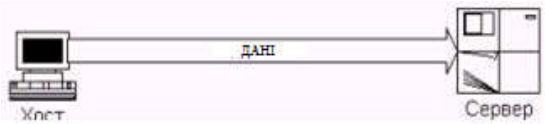

Рисунок 1.5 – Хост передає серверу дані

Перший етап (рисунок 1.6): після закінчення передачі даних хост посилає серверу FIN-пакет із зазначенням власного ISS, ACK і встановленими прапорами FIN і ACK.

Рисунок 1.6 – Перший етап закриття з'єднання

Другий етап (рисунок 1.7): сервер, прийнявши FIN-пакет, посилає хосту підтвердження про прийом.

Рисунок 1.7 – Другий етап закриття з'єднання

Третій етап (рисунок 1.8): сервер посилає хосту FIN-пакет із зазначенням власних даних.

Рисунок 1.8 – Третій етап закриття з'єднання

Четвертий етап (рисунок 1.9): хост передає серверу підтвердження про прийом FIN-пакета. Після цього з'єднання між сервером і хостом буде закрито.

Рисунок 1.9 –Четвертий етап закриття з'єднання

FIN-пакети здатні обійти засоби захисту мережі. Ідея полягає в тому, що який прибув на закритий порт FIN-пакет сервер повинен відповісти RST-пакетом (TCP-пакет з встановленим в ньому прапором RST). FIN-пакети на відкриті порти ігноруються сервером.

Аналогічний прийом використовується при Xmas і NULL-скануванні, за винятком того, що в першому випадку посилає скануючі хосту пакети встановлені з прапором FIN | URG | PSH (звідси назва - «новорічна ялинка»), в той час як у другому випадку будь-які прапори в пакеті відсутні.

Зауважимо, що не всі типи ОС слідують рекомендації, і тому даний метод до них непридатний. Так, ОС Windows 95/98 / NT, по всій видимості, мають

імунітет до такого сканування, проте більшість ОС є сприйнятливими. Таким чином, разом використовуючи SYN і FIN-сканування можна з успіхом обійти засоби захисту сервера і просканувати його порти.

Методи сканування UDP-портів:

1. Сканування UDP-портів перевіркою ICMP-повідомлення «Порт недосяжний».

Цей метод також призначений для визначення стану портів сервера. Основною відмінністю є використання протоколу UDP замість протоколу TCP. Не дивлячись на те, що організація протоколу UDP простіше, ніж TCP, сканувати UDP-порти набагато важче. Це пов'язано перш за все з концепцією протоколу UDP як протоколу з негарантованої доставкою даних. Тому UDP-порт не посилає підтвердження прийому запиту на встановлення з'єднання, і немає ніякої гарантії, що відправлені UDP-порту дані успішно дійдуть до нього.

На щастя, більшість серверів у відповідь на пакет, який прибув на закритий UDP-порт, відправляють ICMP-повідомлення «Порт недоступний» (Port Unreachable - PU). Таким чином, якщо у відповідь на UDP-пакет прийшло ICMP-повідомлення «PU», то сканований порт є закритим, в іншому випадку (при відсутності «PU») порт відкритий. Оскільки немає гарантії, що запити хоста дійдуть до сервера, користувач повинен подбати про повторну передачу UDP-пакета, який, по всій видимості, виявився втраченим.

Цей метод працює дуже повільно через використання на деяких машинах так званої «Компенсації», що обмежує частоту генерування ICMP-повідомлень про помилку. Наприклад, ядро Linux обмежує частоту генерування ICMP-повідомлення «адресат недосяжний» (Destination Unreachable) до 80 повідомлень за 4 секунди, з простоєм ¼ секунди, якщо це обмеження було перевищено. Крім того, для використання даного методу (а саме - для виявлення ICMP-повідомлень про помилку) користувач повинен мати статус root на хості, з якого виробляється сканування.

2. Сканування UDP-портів з використанням функцій recvfrom () і write()

Цей метод використовується в разі, коли користувач, який проводить сканування, не володіє статусом root на хості. Оскільки не-root користувач не

може «читати» ICMP-повідомлення PU, в ОС, що підтримують механізм сокетів (наприклад в Linux), є можливість отримання інформації про стан UDP-порта непрямим способом. Так, наприклад, спроба виклику функції write () на закритий порт зазвичай призводить до виникнення помилки.

Функція recvfrom () в цьому плані більш інформативна. Виклик її на неблокування UDP-сокет сервера зазвичай повертає помилку EAGAIN (Try Again - «спробуйте ще раз», код 13) у разі, коли ICMP-повідомлення не було прийнято, і ECONREFUSED (Connection Refused - «з'єднання закрито», код 111) , якщо ICMP-повідомлення було прийнято.

Може здатися, що сканування UDP-портів не дає тієї повноти інформації, яку можна отримати при скануванні TCP-портів. Однак, беручи до уваги існуючі (хоча і не численні) «дірки» в службах, що використовують протокол UDP (наприклад, «дірку» в rpcbind-демона ОС Solaris, який може знаходиться на будь-якому UDP-порту з номером вище 32770), сканування UDP -Порт здається не таким вже безглуздим.

Таким чином, з усіх перелічених ознаками можливо визначити стан портів скануючого сервера. Найбільша ефективність досягається при використанні комплексного методу сканування, що передбачає вибір конкретного методу або їх сукупності залежно від конкретної ситуації. Як правило, для сканування захищених мереж необхідно використання цілого ряду методів, тоді як для сканування слабо захищених хостів досить одного, причому самого примітивного, на зразок connect ().

Можливості RuNmap. До виникнення ідеї про створення програми Nmap були досліджені можливості багатьох сканерів, таких, як strobe (автор — Julian Assange), netcat (Hobbit), stcp (Uriel Maimon), pscan (Pulvius) ident-scan (Dave Goldsmith) і Satan (Wietse Venema). Всі вони — чудові сканери. Спочатку були спроби доопрацювати код одних сканерів для підтримки кращих можливостей інших.

Потім було вирішено написати абсолютно новий сканер, який використовував б кращі можливості його попередників, і, звичайно, мав нові, такі, як «фрагментоване» сканування та ін. Так вийшов мережевий сканер Nmap – the

Network Mapper. Програма RuNmap являє собою русифіковану та доопрацьовану версію Nmap. Нижче наведені найбільш характерні її можливості.

Динамічне обчислення часу затримки. Деяким сканерів для роботи необхідно вказати час затримки між передачею двох пакетів. Природно, можна використовувати дані ping-запит, але це займе досить багато часу, і, крім того, час затримки постійно змінюється і залежить від «завантаженості» хоста, стану мережі і т. д. RuNmap самостійно визначає час затримки, і підлаштовує його в процесі сканування.

Для root-користувачів використовується найкращий спосіб визначення часу затримки – функція ping. Для інших цей параметр визначається функцією connect () на закритий порт. Крім того, у користувача є можливість самостійно встановити час затримки, але зазвичай робити це немає необхідності.

Повторна передача пакетів. Деякі сканери відразу відправляють всі запити, а потім «збирають» відповіді на них. Дуже некоректний підхід, оскільки при цьому не береться до уваги той факт, що іноді пакети можуть просто не дійти до адресата по самим різним причинам. У такій ситуації сканер прийме рішення про відсутність відповіді і помилково вкаже, що сканований порт закритий. Найбільше помилок виникає при використанні «негативного» сканування типу UDP або FIN, рішення яких приймається на основі відсутності відповіді. RuNmap автоматично вибирає число повторних передач пакетів на порти, від яких не було отримано відповідь.

Паралельне сканування портів. Деякі сканери послідовно сканують всі 65535 портів за один раз. Цей метод нормально працює лише при скануванні TCP-портів в високошвидкісних локальних мережах. Глобальні мережі типу Internet високою швидкістю не відрізняються. RuNmap використовує неблокированный введення/виведення (non-blocked i/o) і паралельне сканування у всіх режимах TCP і UDP. Ви можете задати число паралельних процесів сканування самостійно. На дуже швидких мережах ефективність сканування зменшується при вказівці цього значення більше 18. На повільних мережах — навпаки, чим більше значення, тим вище ефективність.

Гнучке вказівку сканованих портів. Часто буває необхідно відсканувати які-небудь конкретні порти, а не всі 65535 портів відразу. Більшість сканерів дозволяє задавати діапазон портів типу 1-N, що також не завжди прийнятно. RuNmap дозволяє задати будь-яку кількість довільних діапазонів і портів, наприклад ’21-25,80,113,6000-‘.

Гнучке завдання мети сканування. Часто необхідно просканувати більш ніж один хост, однак більшість сканерів дозволяють задати лише одна адреса. Все, що не є опцією або її аргументом, RuNmap сприймає як адреса хоста. Таким чином, ви можете абсолютно довільно вказати адреси та діапазони адрес, які потрібно просканувати.

Визначення неактивних хостів. Деякі сканери дозволяють сканувати великі мережі, проте вони витрачають дуже багато часу на сканування 65535 портів хоста, який з якихось причин не функціонує. За замовчуванням, перед скануванням RuNmap опитує кожен хост і визначає його стан, для того, щоб не витрачати часу на сканування неактивних хостів.

Визначення IP-адреси скануючого хоста. З деяких причин більшість сканерів вимагають вказати використовуваний скануючим хостом IP-адресу в якості одного з параметрів. RuNmap автоматично визначає IP-адресу машини, на якій він працює, на стадії ping-опитування, і використовує ту адресу, на який надійшла відповідь. Якщо він не зміг це зробити (наприклад, користувач відключив ping-опитування), RuNmap пробує визначити первинний мережевий інтерфейс і використовує його IP-адресу. Нарешті, користувач сам може вказати IP-адресу хоста.

1.2 Програмне забезпечення для сканування портів

На даному етапі роботи було проаналізовано декілька програмних утиліт за допомогою яких можна просканувати комп’ютерну мережу на наявність відкритих портів. Короткий опис цих програм представлено нижче.

Port Scanner by ENTER – це компактний, швидкий, повністю безкоштовний, надійний і простий у використанні сканер портів для платформи Win32 скануючий локальний комп'ютер або будь-який хост на наявність відкритих портів, яким зможуть скористатися для проведення аудиту чи інших цілей.

Він вживає технологію багатопотокового (multithread), тому на потужних комп'ютерах ви можете сканувати порти з дуже високою швидкістю.

Також він містить опис стандартних портів і може виконувати сканування у передбаченому діапазоні портів.

Сканування портів (Port Scanning) може бути одним з перших кроків для злому або захисту від злому мережі, визначаючи основну мету атаки. За допомогою сканера портів можна досліджень служби, встановлені на комп'ютері (FTP-сервер, Web-сервер, mail-сервер,).

Можливості:

– дозволяє сканувати групу портів, проміжок або окремий порт;

– можливість працювати в фоні;

– спливаючі вікна після закінчення обробки;

– збереження результату в файл;

– перегляд поточного дії в реальному часі.

NetFlow Traffic Analyzer (NTA) дозволяє записувати дані з безперервного потоку мережевого трафіку їх аналізувати і складати графіки і таблиці, за допомогою яких можна переглянути, ким і з якою метою використовувалася корпоративна мережа з моніторингом CBQoS ви можете призначити потрібні пріоритети трафіку в мережі.

За допомогою NetFlow Traffic Analyzer ви можете отримати повне уявлення про мережу трафіку, знайти "вузькі" місця і обмежити пропускну здатність недобропорядним співробітникам.

Можливості Orion NetFlow Traffic Analyzer:

– визначає, які користувачі, додатки та протоколи споживають найбільше мережевий пропускної здатності і вказує IP адреси найбільш активних користувачів в мережі;

– моніторинг мережевого трафіку шляхом захоплення потоку даних від мережевих пристроїв, в тому числі Cisco ® NetFlow v5 або v9, Juniper ® J-Flow, IPFIX і SFlow;

– складає карти трафіку надходять від призначених портів, в яких вказує: IP-адреси джерела, призначення IP-адрес, і навіть протоколи, щоб можна було дізнатися імена додатків;

– забезпечує миттєве повідомлення адміністратора про перевищення порога використання пропускної здатності;

– за допомогою Class-Based Quality of Service (CBQoS) ви можете призначати необхідний рівень пріоритету трафіку;

– генерує звіти мережевого трафіку за допомогою всього лише кількох кліків.

Port Explorer- це аналізатор сокетів і розвідка, призначена як для новачків, так і для просунутих користувачів. Окрім потужного картографічного порту для обробки, порт Explorer також озброєний розширеними наборами пакетів, виявленням країни, анти-троянськими можливостями та дроселюванням пропускної здатності, а також міні-арсеналом підтримуючих мережевих інструментів, включаючи повністю автоматичний Whois клієнт пошуку Port Explorer дає вам змогу бачити всі відкриті сокети та показує стан, в якому вони знаходяться, якщо воно встановлено, а також надсилати чи отримувати дані, слухати або закривати.

Port Explorer також дає чудову можливість контролювати дані сокети, дозволяючи користувачеві шпигуватис за будь-якими або всіма сокетами, що належать процесові, і блокувати надсилання або отримання будь-якого або всіх розеток.

Користувач може блокувати надсилання даних через підозрілу розетку, але все одно бачить, які дані вводять шпигунством на цьому сокеті. Port Explorer також включає в себе нове виявлення троянів, показуючи приховані сокети червоним кольором.

1.3 Способи сканування портів та механізми виявлення мережевих атак

Способи сканування.

Горизонтальне сканування. При горизонтальному скануванні зловмисник переглядає один і той же порт на декількох комп'ютерах, то є кілька IP-адрес. Атакуючий прагне знайти хости, що розкривають певні сервіси. Таким чином хакер сканує певні порти на всіх машинах, різні IP-адреси в межах певного діапазону. Горизонтальне сканування є найбільш часто використовуваний в даний момент тип сканування портів.

Вертикальне сканування. Вертикальним скануванням називають процес, при якому зловмисник сканує кілька портів на одному комп'ютері, тобто один IP-адрес.

Розподілене вертикальне сканування. При розподіленому вертикальному скануванні кілька джерел послідовно сканують кілька портів на одному IP-адресу.

Розподілене горизонтальне сканування. У цьому випадку кілька джерел сканують один і той же порт на декількох IP-адресах послідовним чином. Під час розподіленого сканування часто змінюються IP-адреси зловмисників, що робить їх виявлення досить складним завданням.

Розподілене вертикальне сканування і розподілене горизонтальне сканування часто асоціюються з атаками, виконуваними кількома зловмисниками (або групами), що представляє собою одну з найсучасніших форм атак. Спільні атаки іноді описуються як «кібератаки наступного покоління». Нещодавно опубліковане дослідження представило новий підхід, який допомагає виявити сканування портів, він заснований на графічному моделюванні. Творці нового підходу застосували свій метод до даних, отриманих з darknet. Схему сканування портів показано на рисунку 1.10.

Спочатку системні адміністратори могли визначити вторгнення, сидячи перед консоллю і аналізуючи дії користувачів, звернувши увагу на аномальну активність користувача, який будучи у відпустці локально увійшов в систему. Або звернути увагу на незвично високу активність принтера, який

використовується рідко. Колись така форма виявлення була досить ефективною, і орієнтувалася на конкретні дії. Вона не мала масштабу. Наступним кроком по виявленню зловмисних дій стало введення журналів реєстрації, і їх аналіз. Так як аналіз, на предмет ознак незвичайних дій, здійснювався візуальним переглядом величезних куп перфорованого паперу (в кінці 70-х - 80-х журнали реєстрації друкувалися на перфорованому папері), займав багато часу. Як правило системні адміністратори використовували журнали реєстрації, після звершення будь-яких зловмисних дій, щоб довести факт їх здійснення. Ймовірність виявлення вторгнень під час їх здійснення була вкрай мала, це було пов'язано з виключно ручними методами аналізу при досить великій кількості інформації.

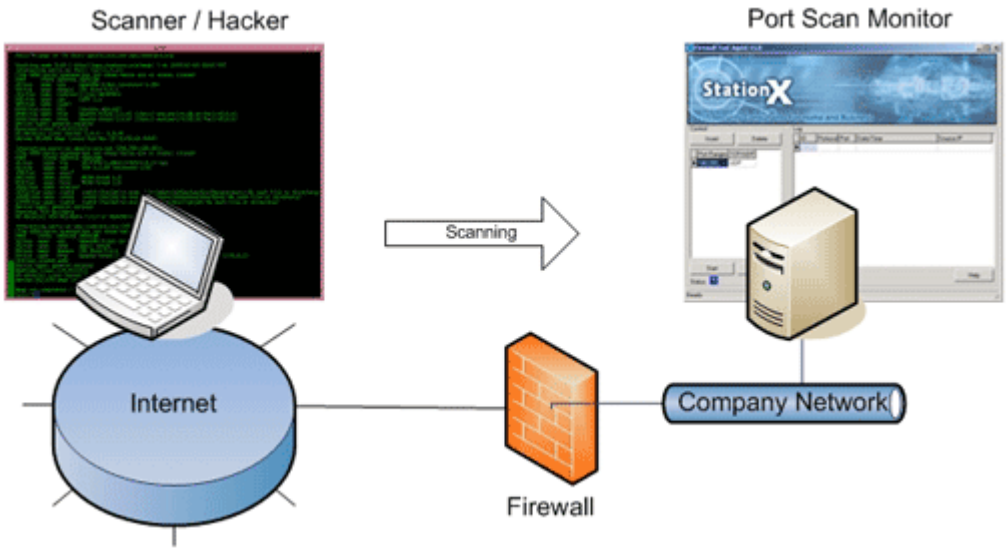

Рисунок 1.10 – Схема сканування портів

У міру зростання технічних характеристик комп'ютерів (дискова пам'ять, продуктивність), журнали реєстрації стали доступні в електронному вигляді, що послужило появі програмних засобів, для аналізу протоколювання даних. Дані програмні засоби вимагали значної обчислювальної потужності і тому працювали досить повільно. Найчастіше їх запускали в пакетному режимі, в кінці робочого дня і залишали на ніч, тому що в цей час активність користувачів була найменшою. На даному етапі розвитку більшість зловмисних дій визначалася як і раніше після їх звершення.

У дев'яностих роках почали розроблятися СОВ, що працюють в оперативному режимі, аналізують тільки що згенеровані записи в журналах реєстрації. Дані СОВ дозволяли виявляти вторгнення в момент їх здійснення, дозволяючи вжити відповідних заходів, а іноді визначати можливість вторгнення до їх здійснення. На даний момент всі проекти, пов'язані з пошуком і виявленням вторгнень, проектуються під ефективну роботу в великих мережах, і є складним завданням, тому що весь час зростає число нових методів здійснення вторгнень і безперервні зміни в технічному середовищі.

Методи виявлення. Аналіз отриманої інформації вкрай важлива характеристика будь-якої системи виявлення вторгнень. Існує два основні методи виявлення зловмисних дій: виявлення аномалій і виявлення зловживань.

Виявлення аномалій. Виявлення аномалій використовує моделі передбачуваного поведінки користувачів і додатків, інтерпретуючи відхилення від «нормальної» поведінки як потенційне порушення захисту.

Методи виявлення аномалій засновані на тому, що дії при вторгненні відрізняються від нормальної поведінки системи. При визначенні аномалій створюються профілі, що описують нормальну роботу користувачів, хостів або мережевих з'єднань, які можна змоделювати досить точно. Наприклад, конкретний користувач виконує звичайні повсякденні дії (реєструється в системі в певний час, переглядає електронну пошту, виконує транзакції баз даних, не проявляє робочу активність в обід і після закінчення робочого дня, допускає незначну кількість помилок при доступі до файлів і так далі). Дані профілі створюються, на основі даних історії під час нормального функціонування системи. Далі, якщо система відзначає, що користувач виконує дії не відповідають записаному профілем (реєстрація в системі в нічний час, використання засобу компіляції та від лагодження, відбувається велика кількість помилок при доступі до файлів), і визначає таку діяльність як підозрілу.

На жаль, методи виявлення аномалій і СОВ, засновані на них, формують багато хибних тривог, так як зразки з описом «нормального» поведінки користувача і системи не мають чітких меж. Незважаючи на цей недолік, є ймовірність, що системи, засновані на визначенні аномалій, зможуть визначати

нові форми атак. До того ж такі системи важко настроюються в середовищі, де характерна значна мінливість. Виявлення зловживань. Методи виявлення зловживань припускають, що аналізуючи діяльність системи, події або їх безлічі проводиться звірка на відповідність із заздалегідь визначеними зразками, що описують відомі атаки. Якщо порівнюваний зразок відповідає відомій атаці, то така відповідність називається сигнатурою.

Використання методів визначення зловживань досить ефективно для визначення атак, при цьому не створюється багато неправдивих повідомлень (тривог). При визначенні зловживань діагностуються кошти або технології атаки досить швидко, що дозволяє адміністратору своєчасно скоригувати заходи забезпечення безпеки. Вони дозволяють адміністраторам системи, незалежно від рівня їх кваліфікації в галузі безпеки, проаналізувати інцидент.

Недоліки цього підходу є постійний контроль за оновленням баз даних з новими сигнатурами, так як детектори можуть провести аналіз тільки по заздалегідь відомим сигнатурам.

Після отримання та аналізу інформації про подію, система здійснює відповідні дії, що представляють собою звіти, створені в стандартному форматі, або автоматизовані дії (блокування порту, програма). Існують і більш активні відповіді (автоматизовані діями), що виконуються при визначенні конкретних типів атак, наприклад відправка повідомлення на телефон, включення попереджувального звукового сигналу або організація контратаки.

Активні дії можуть включати в себе перенастроювання конфігурації роутера з метою блокування адреси атакуючого або організувати відповідний напад на зловмисника. Наприклад, при вторгненні з використанням фальсифікованого трафіку з підставним IP-адресою, система може змінити конфігурацію роутерів і блокувати одержуваний трафік з даної адреси, щобуде означати організацію атаки «відмова в обслуговуванні» для хоста, за який видає себе хакер.

Успішне сканування портів дозволяє отримати відомості про машини, виявлених в мережі, включаючи імена пристроїв, IP-адреси, операційні системи, програмне забезпечення та служби, імена користувачів, групи та відкриті порти. Сканування звичайно виконують до початку атаки.

2 АЛГОРИТМИ ВИЯВЛЕННЯ СКАНУВАННЯ ПОРТІВ В КОМПЮТЕРНІЙ МЕРЕЖІ

2.1 Опис алгоритму системи Port Scan Attack Detector

Інструментальний комплект Port Scan Attack Detector (PSAD) PSAD представляє собою набір з трьох простих, але ефективних системних демонів (двох основних та одного допоміжного), що виконують оперативний аналіз повідомлень від iptables в системних журналах (логах). Саме ця комбінація iptables і PSAD працює найбільш швидко і ефективно при аналізі мережевого трафіку.

Відразу слід зазначити, що PSAD забезпечує підтримку аналізу логів як IPv4, так і IPv6, що генеруються в iptables та ip6tables відповідно.Наочно роботу PSAD можна побачити на рисунку 2.1

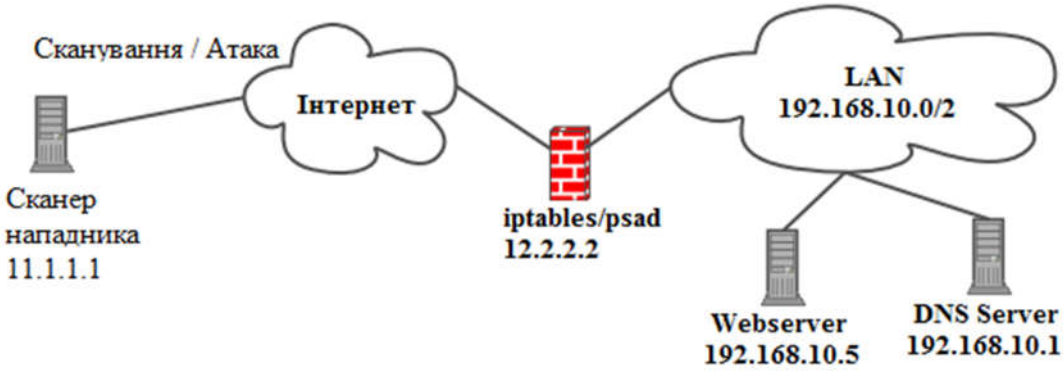

Рисунок 2.1 – Структура роботи PSAD

З більш широкомасштабної комплексної системи виявлення зовнішніх вторгнень Snort в PSAD включає безліч підписів (правил), що дозволяють виявити спроби введення різних троянських програм (EvilFTP, GirlFriend, SubSeven та ін.), спроби проведення DDoS-атак засобами mstream, shaft і такими поділами, а також більш ізольоване, замасковане сканування портів (FIN, NULL, XMAS), виконується за допомогою nmap. Крім того, забезпечується виявлення сканування портів TCP-сегментами типу SYN та UDP-пакетами шляхом ретельно дослідження полів заголовків різних мережних пакетів. В поєднанні з парсером

правил fwsnort та додатковим засобом для Netfilter, що дозволяє виконувати пошук узгоджень строк, він може виявити більшість описаних у наборі правил Snort атак, орієнтованих на рівень додатків та їх даних. Так само передбачена перевірка полів заголовка ICMP-повідомлень (зокрема, перевіряються поля типу і коди).

Перевірка вмісту мережевих пакетів дозволяє виявити атаки типу "переповнення буфера" (буферний переповнення), потенційно небезпечні команди на рівні рівня та інші небезпечні елементи трафіку, знову ж таки, за допомогою парсера та підсистеми пошуку стрічок.

При необхідності сповіщення про виявлені факти атак та спроби вторгнення відправляються по електронній пошті, зазначеній при конфігуруванні. У таких повідомленнях можуть міститися детальні характеристики TCP / UDP / ICMP-сканування, дані про реверс DNS і хто, виявлені збіги з правилами Snort та інша важлива інформація.

У PSAD є можливість моніторингу набору правил iptables з метою перевірки дотримання основного правила безпеки політики "за замовчуванням падіння" (тобто є пакет, що не відповідає ні одному правилу iptables, за замовчуванням не приймається, відкидається).

При конфігурації PSAD можна визначити різні "порогові" (порогові) або граничні значення параметрів, що впливають на виявлення сканування, а також встановлювати загальний рівень допустимих мережевих небезпек (рівень небезпеки), який може бути присвоєно автоматично і жорстко зафіксований без можливості зміни для конкретних підмереж і / або IP-вузлів.

Відповідно до встановлених при налаштуванні порогових значень параметрів виявлення сканування та рівня мережевої небезпеки може бути виконана автоматична блокування IP-адрес, з яких здійснюється сканування портів. Це робиться засобами iptables / ip6tables та / або tcpwrappers. Потрібно звернути особливу увагу на те, що з самого початку за замовчуванням блокування функція відключене, і потрібна її активізація при виникненні такої необхідності.

Окрім всього вищесказаного в PSAD передбачено режим спеціального режиму статусу, в якому виведено докладну інформацію про поточний стан

спостереження за спробами сканування з відповідними пакунками лічильників, правилами послідовності iptables і ip6tables, рівнями небезпеки та іншими корисними характеристиками.

Особливості:

− виявлення для сканування TCP, SYN, FIN, NULL, XMAS, а також сканування UDP;

− підтримка журналів IPv4 та IPv6, створених iptables та ip6tables відповідно;

− виявлення багатьох правил підписів з системи, виявлення вторгнення Snort;

− тестовий аналіз iptables та ip6tables logfile analysis (корисний як криміналістичний інструмент для вилучення інформації сканування з старих лог файлів iptables і ip6tables);

− пасивна робота операційної системи через пакети синтезу TCP. Підтримуються дві різні стратегії відбитків пальців; повторна реалізація p0f, яка суворо використовує повідомлення журналу iptables та ip6tables (вимагає командного рядка -log-tcp-options) і стратегії на основі TOS;

− електронні сповіщення, що містять характеристики сканування TCP, UDP, ICMP, зворотні дані DNS та Whois збій правил, віддалені відомості про ОС, і багато іншого;

− у поєднанні з fwsnort і iptables розширення string match, PSAD може генерувати сповіщення для атак переповнення буфера додатків, підозрілі заявки команд;

− Icmp тип та перевірка поля заголовка коду;

− настроюванні пороги сканування та призначення рівня небезпеки;

− Iptables налаштовує синтаксичний аналіз, щоб перевірити позицію політики "за умовчанням";

− автоматичне присвоєння IP мережевої небезпеки (може використовуватися для ігнорування або автоматичного збільшення рівнів небезпеки для певних мереж);

− дзвінки DShield;

− автоблокування скануючих IP-адрес за допомогою iptables і ip6tables або tcpwrappers на основі рівня небезпеки сканування. (Ця функція НЕ активована за замовчуванням.);

− аналіз журнальних повідомлень iptables і ip6tables та генерація виходу CSV, який може бути використаний як вхід для AfterGlow. Це дозволяє візуалізувати журнали iptables та ip6tables. Gnuplot також підтримується;

− режим стану, який показує підсумок поточної інформації про сканування з відповідними значеннями пакетів, ланцюгами iptables і ip6tables та рівнем небезпеки.

Весь аналіз даних PSAD збирається з повідомлень журналу iptables. PSAD за замовчуванням читає файл / var / log / messages для нових повідомлень iptables і, записує їх у спеціальний файл (/ var / log / PSAD / fwdata). PSAD відповідає за застосування порогів небезпеки та логіки підписів, щоб визначити, чи було виконано сканування порту, надсилати відповідні електронні повідомлення(необов'язково) та блокувати порушення IP-адрес. PSAD включає в себе обробник сигналів, такий, що якщо прийматиметься сигнал USR1, PSAD скине вміст поточної структури даних хешу сканування до /var/log/PSAD/scan_hash.$$, де "$$" репрезентує підставу робочого демона PSAD.

Залежно від дистрибутива Linux, PSAD може бути вже доступним в репозиторії пакетів за замовчуванням. Наприклад, на системах Debian або Ubuntu, установка виконується просто:

```
apt-get install PSAD
```

Якщо PSAD не доступний у сховищі пакетів, його можна встановити за допомогою сценарію install.pl, що входить до складу джерел PSAD. Сценарій install.pl також обробляє оновлення, якщо PSAD вже встановлено. PSAD вимагає декілька модулів perl, які можуть або не можуть бути вже встановлені у вашій системі Linux. Ці модулі входять до каталогу deps / directory в джерелах PSAD, і вони автоматично встановлюються сценарієм install.pl. Список модулів:

− Bit::Vector;

− Date::Calc;

− IPTables::ChainMgr;

− IPTables::Parse;

− NetAddr::IP;

− Storable;

− Unix::Syslog.

Установка та конфігурація PSAD. Перед установкою PSAD слід уточнити декілька ключових моментів і, якщо це необхідно, виконати кілька нескладних, але дуже важливих операцій.

По-перше, як вже було сказано раніше, PSAD використовує повідомлення, що генеруються iptables та ip6tables, в яких міститься інформація про вжиті або пропущені Інтернет пакетах. Тому конфігурація мережевого екрану (firewall) повинна передбачати запис даних про мережеві пакетах в системний журнал (log), інакше PSAD просто не зможе виявляти спроби сканування портів і виконувати інші дії щодо захисту системи.

Власне кажучи, для організації журналювання пакетів потрібно додати всього лише два простих правила:

```
iptables -A INPUT -j LOG
iptables -A FORWARD -j LOG
```

Проте, на різних системах можуть виникати додаткові вимоги до правил iptables, тому рекомендується уважно вивчити приклади наборів правил для мережевого екрану, повністю сумісних з PSAD, які містяться в файлі FW_EXAMPLE_RULES, що входить до складу дистрибутивного пакету PSAD. При цьому потрібно врахувати, що PSAD не сумісний з застарілими мережевими екранами ipchains і ipfw, підтримуваними linux-ядрами версій 2.4.x і більш старими.

По-друге, необхідно перевірити розділ конфігурації (config section) скрипта установки install.pl (він написаний на мові Perl). Втім, це зауваження відноситься тільки до варіанту установки з tarball-пакета, що містить вихідні коди на C і Perl.

У розділі конфігурації потрібно звернути увагу на префікси шляхів установки PSAD - за замовчуванням демони PSAD, kmsgsd і PSADwatchd встановлюються в каталог / usr / sbin, але це можна змінити, а також перевірити правильність зазначених шляхів доступу до системних утиліт (chkconfig, make, perl, wget, runlevel), необхідних для коректної установки PSAD.

Сам процес установки можна виконати за допомогою менеджера пакетів для встановленого на даній конкретній системі дистрибутива (для Debian apt-get, для Fedora yum, для OpenSUSE yast і т.д.) або з дистрибутивного tarball-пакета з вихідними кодами. В останньому випадку після розпакування пакету потрібно перейти в основний каталог вихідних кодів PSAD, отримати повноваження root і запустити скрипт установки:

```
# ./install.pl
```

Якщо в процесі установки буде виявлена більш рання версія PSAD, встановлена в системі, то стару конфігурацію можна зберегти, відповівши ствердно на відповідне запитання програми. У тому випадку, коли заздалегідь приймається рішення про те, що попередня версія PSAD і її конфігурація не потрібні, можна дати команду установки наступним чином:

```
# ./Install.pl –n
```

Після успішно завершеного процесу установки слід виконати команду (також з привілеями root):

```
# /Etc/init.d/PSAD-init start
```

Для активізації демонів PSAD, kmsgsd і PSADwatchd або просто запустити їх по черзі з командного рядка.

Повне видалення PSAD і всіх його компонентів з системи виконується командою:

./Install.pl --uninstall

Конфігурування. Для нормальної роботи PSAD необхідно, щоб всі повідомлення kern.info записувалися в іменований програмний канал / var / lib / PSAD / PSADfifo. Тому в файл конфігурації /etc/syslog.conf або в деяких системах / etc / sysconfig / syslog потрібно додати наступний рядок:

kern.info | / var / lib / PSAD / PSADfifo

(Kern.info і символ '|' розділені табуляцією, а не пробілами).

Після внесення зміни і збереження файлу конфігурації потрібно перезапуск сервісу syslog:

/Etc/init.d/syslog restart

або в деяких системах:

/Etc/init.d/sysklogd restart
/etc/init.d/klogd

Параметри конфігурації PSAD розміщені в чотирьох файлах /etc/PSAD/PSAD.conf, /etc/PSAD/fw_search.conf, /etc/PSAD/kmsgsd.conf і /etc/PSAD/PSADwatchd.conf. Всі записи в цих файлах виконуються за схемою "ключ / значення", тобто, спочатку записується ім'я параметра, одночасно служить ключем, потім слідує значення даного параметра. Значення супроводжує обов'язковий завершальний символ - крапка з комою. Якщо рядок починається з символу '#', то вона інтерпретується як коментар. Коментарі можна записувати і в кінці рядків, після крапки з комою, завершальній значення параметра.

Після установки більшістю параметрів присвоюються значення за замовчуванням. Зрозуміло, всі параметри можуть бути змінені в будь-який час. У

наступному розділі деякі найбільш важливі параметри конфігурації розглядаються більш докладно.

Деякі параметри конфігурації і їх допустимі значення.

Описувані нижче параметри визначаються у файлі конфігурації /etc/PSAD/PSAD.conf, якщо явно не вказано інше ім'я файлу.

Параметр EMAIL_ADDRESSES визначає адреси, за якими PSAD повинен відправляти оповіщення, що попереджають про спроби сканування портів та інших атаках, а також повідомлення про поточний стан. У списку email-адрес роздільником служить кома. За замовчуванням визначається адреса root @ localhost, але вже при установці PSAD користувачеві видається запит на зміну цієї адреси. Приклад визначення списку email-адрес:

EMAIL_ADDRESSES ladmin @ myhome.net, boss @ localhost;

Для параметра HOSTNAME вказується повне ім'я комп'ютера із зазначенням домену (FQDN), наприклад:

HOSTNAME desktop.myhome.net

Параметр HOME_NET визначає підмережа, що є областю дії PSAD. Ця змінна використовується для ідентифікації трафіку, що збігається з snort-правилами в ланцюжках FORWARD мережевого екрану iptables. Трафік, що регулюється ланцюжками INPUT і OUTPUT, за замовчуванням вважається відповідно вхідний і вихідний щодо HOME_NET, отже, навіть якщо цей параметр не визначений, PSAD зможе виявляти збіги з правилами в трафіку. Проте, якщо змінна HOME_NET не визначена, то генеруються попереджуючі повідомлення в syslog і для відправки по email. Визначення підмереж записуються в форматі безкласової междоменной маршрутизації / адресації (CIDR) і розділяються комами, якщо в списку міститься більше однієї мережі. Приклад визначення даного параметра для простої домашньої підмережі:HOME_NET 192.168.10.1/24;

Якщо в системі користувача є тільки один мережевий інтерфейс (тобто, трафік по ланцюжку FORWARD проходити не буде через відсутність мети), то параметр HOME_NET фактично не використовується, але його треба визначити наступним чином:

HOME_NET NOT_USED;

Параметр SYSLOG_DAEMON встановлює тип використовуваного демона, що виконує запис повідомлень в системні журнали (логи). PSAD підтримує три різні реалізації: syslogd, syslog-ng і metalog. За замовчуванням використовується syslogd:

SYSLOG_DAEMON syslogd;

Параметр DANGER_LEVEL {n} - рівень небезпеки (тут n може мати числове значення від 1 до 5) - встановлює граничну кількість пакетів, які повинні бути виявлені і кваліфіковані, як підозрілі, для того, щоб констатувати факт досягнення відповідного рівня небезпеки. Рівень небезпеки 1 передбачає мінімум граничної кількості підозрілих пакетів, а на рівні небезпеки 5 допускається найбільша кількість виявлених підозрілих пакетів. Після визначення рівнів небезпеки їх можна призначати для випадків збігу з окремими сигнатурами-зразками (файл / etc / PSAD / signatures) і для конкретних випадків (варіантів) сканування портів (див. Файл / etc / PSAD / auto_dl). У лістингу 1 показаний приклад визначення рівнів небезпеки, приймається за замовчуванням.

Лістинг 1. Визначення рівнів небезпеки

1 DANGER_LEVEL1 5;
2 DANGER_LEVEL2 15;
3 DANGER_LEVEL3 150;
4 DANGER_LEVEL4 1500;
5 DANGER_LEVEL5 10000;

Параметр PSAD_CHECK_INTERVAL визначає інтервал в секундах, на який PSAD "засинає", перш ніж виконати чергову перевірку повідомлень від iptables в системних журналах. За замовчуванням заданий інтервал в 5 секунд:

PSAD_CHECK_INTERVAL 5;

Параметр IGNORE_PORTS визначає список TCP і / або UDP портів, які PSAD перевіряти не повинен при будь-яких умовах, навіть якщо на цих портах реєструється підозрілий трафік. Номери портів в списку розділяються комами, можуть бути задані як окремі порти, так і діапазони портів. За замовчуванням PSAD перевіряє всі порти, нічого не ігноруючи:

IGNORE_PORTS NONE;

Але при необхідності можна скасувати перевірку деяких портів:

IGNORE_PORTS tcp / 61000-61356, udp / 53, udp / 5000;

Параметр ENABLE_AUTO_IDS є перемикач дозволу / заборони автоматичного блокування IP-адрес, з яких виробляється сканування портів або виходить який-небудь інший підозрілий трафік. За замовчуванням автоматичне блокування заборонено, щоб виключити проблеми, які можуть виникати при не цілком правильного налаштування політик iptables. При необхідності автоматичне блокування IP-адрес дозволяється заміною значення за замовчуванням 'N' на значення 'Y':

ENABLE_AUTO_IDS Y;

Параметр AUTO_IDS_DANGER_LEVEL встановлює граничне значення рівня небезпеки, на який повинен вийти процес сканування портів, перш ніж PSAD автоматично заблокує атакуючий IP-адреса. Зрозуміло, при цьому повинна

бути дозволена автоматичне блокування IP-адрес. За замовчуванням визначено п'ятий рівень небезпеки:

AUTO_IDS_DANGER_LEVEL 5;

Параметр AUTO_BLOCK_TIMEOUT визначає інтервал часу блокування атакуючого IP-адреси в секундах (автоматичне блокування повинна бути дозволена). За замовчуванням встановлюється блокування тривалістю одна година:

AUTO_BLOCK_TIMEOUT 3600;

Опис роботи деяких функцій PSAD

Для отримання повного звіту про роботу PSAD і про його поточний стан, що включає дані про використання процесора і оперативної пам'яті всіма трьома демонами, а також кількість оброблених пакетів з розкладкою за їхніми джерелами, необхідно отримати повноваження супер root і виконати наступну команду:

PSAD -S

(Або з "довгим" прапором: PSAD --Status).

Висновок цієї команди містить таблицю, в якій зазначено, скільки TCP, UDP і ICMP пакетів обробив PSAD з моменту свого запуску. Слід зазначити, що не всі пакети, повідомлення про які зареєстровані в системних журналах, відносяться до виявленими фактами сканування, тому загальна кількість вхідних пакетів наведено нижче на рисунку 2.2, в розділі "Total packet counters".

Після сканування Nmap статус PSADпоказано на рисунку 2.3 та рисунку 2.4.

Рисунок 2.2 – Результат роботи команди PSAD –S

Примітка: в стовпці dl зазначений рівень небезпеки (danger level), встановлений PSAD на момент виведення таблиці стану.

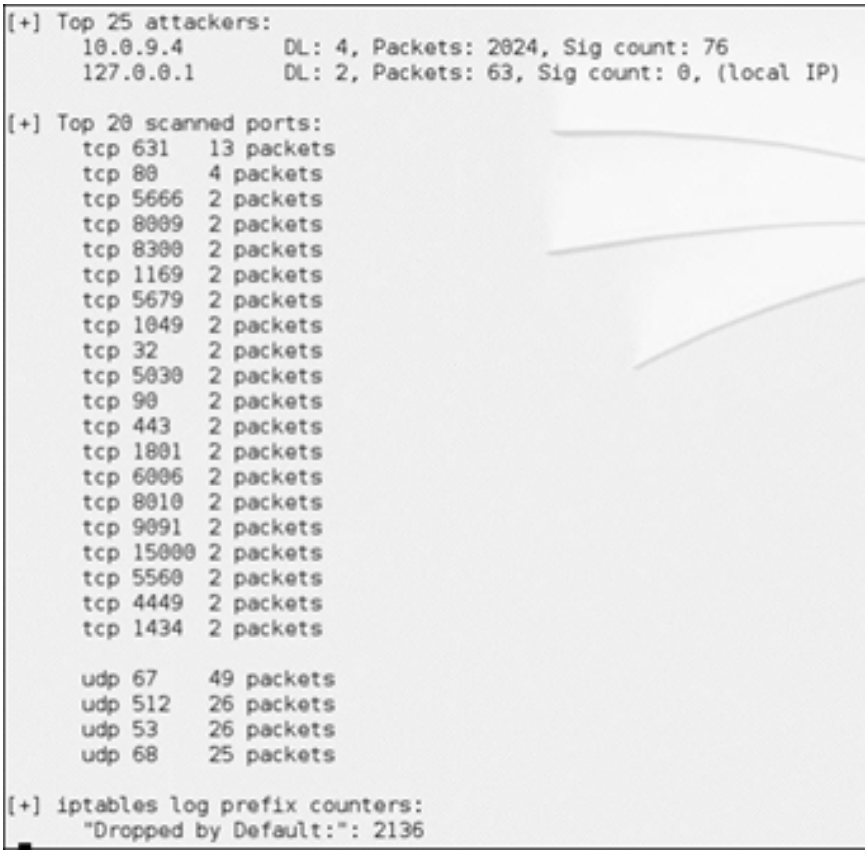

Рисунок 2.3 – Статус PSADпід час атаки

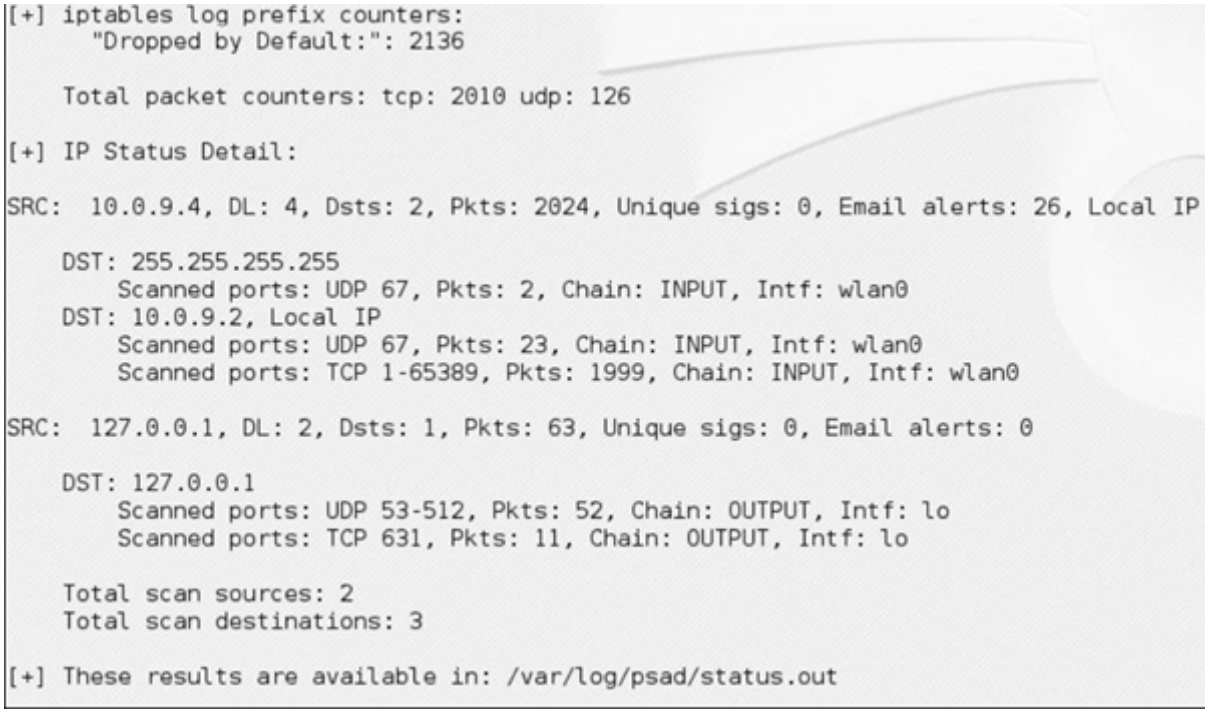

Рисунок 2.4 – Продовження нового статусу PSAD

Отримання більш докладної інформації по конкретному IP-адресу. Дані по окремим IP-адресами можна отримати за допомогою команди:

PSAD --status-ip <ip-адреса>

Тут виводяться стану лічильників пакетів по кожному протоколу і останні 10 пакетів, зареєстровані iptables в системному журналі, для заданої IP-адреси. Але можна отримати і більш докладну інформацію. В черговий раз, як завжди, коли справа стосується безпеки системи, будуть потрібні повноваження супер root. Дані по IP-адресами зберігаються в каталозі / var / log / PSAD. Кожному окремому IP-адресу відповідає свій підкаталог. У лістингу 3 показаний приклад для IP-адреси 11.22.22.33.Для перегляду інформації, що міститься в показаннях вище файлах, можна скористатися системною утилітою less (або cat, якщо файл досить невеликий).

Автоматичне блокування атакуючих IP-адрес. Як вже було зазначено в першій статті циклу, для того, щоб зробити можливим автоматичне блокування особливо небезпечних і активно атакуючих IP-адрес, необхідно встановити для параметра конфігурації ENABLE_AUTO_IDS значення Y (за замовчуванням

блокування відключено). Крім того, необхідно також змінити значення на Y для параметра IPTABLES_BLOCK_METHOD. Цей параметр визначає, щоблокування IP-адрес буде виконуватися за допомогою iptables, що є більш ефективним і більш безпечним методом, ніж, скажімо, блокування за допомогою tcpwrappers, оскільки при iptables-блокуванні пакети перехоплюються на рівні ядра до того, як вони отримують яку б то не було можливість взаємодії з будь-яким демоном або якою-небудь програмою. Таким чином, блокування за допомогою iptables є найкращим метод захисту.Після того, як перераховані вище параметри змінені:

ENABLE_AUTO_IDS Y;
IPTABLES_BLOCK_METHOD Y;

В файлі конфігурації /etc/PSAD/PSAD.conf, і файл збережений, необхідно перезапустити сервіс PSAD, щоб внесені зміни вступили в силу:

/Etc/init.d/PSAD restart

Перезапуск можна виконати і за допомогою команди PSAD -R або PSAD --Restart, яка також дозволяє перезапустити всі PSAD-процеси. Позитивним моментом тут є те, що даний ключ "згадує" і використовує всі параметри командного рядка, з якими був запущений початковий PSAD-процес.Режим автоматичного блокування IP-адрес в будь-який момент можна скасувати командою:

PSAD -F

(Або з використанням "довгого" ключа: PSAD --Flash). При цьому видаляються всі автоматично згенерувані правила блокування iptables.Отримання зведення параметрів конфігурації. Поточний список конфігураційних параметрів PSAD можна отримати, використовуючи команду:

PSAD -D

(Або з використанням "довгого" ключа: PSAD --Dump-conf). Список виводиться на стандартний пристрій виводу. Слід зазначити, що в виведений список не включаються деякі «критичні» параметри, такі як дані про мережі, що захищається (HOME_NET), email-адреси, за якими відправляються попередження, і деякі інші.

2.2 Опис утиліти Snort

Snort - це мережева система виявлення (IDS) та запобігання вторгненню (IPS) з відкритим вихідним кодом, здатна здійснювати реєстрацію пакетів і в реальному часі здійснювати аналіз трафіку в IP-мережах, комбінуючи можливості порівняння за списками, засобами для інспекції протоколів та механізмами виявлення аномалій. Snort був створений Мартіном Решем в 1998-му році і дуже швидко завоював популярність, як безкоштовна система виявлення вторгнення, що дозволяє самостійно і без особливих зусиль писати правила для виявлення атак. По суті, мова опису підписів Snort став стандартом де-факто для багатьох систем виявлення вторгнень, які стали його використовувати в своїх кодах.

Структура і функціонування Snort. Систему виявлення вторгнень Snort за способом моніторингу системи можна віднести як до вузлової, так і до мережевої системи в залежності від параметрів настройки. Зазвичай вона захищає певний сегмент локальної мережі від зовнішніх атак з інтернету. Система Snort виконує протоколювання, аналіз, пошук по вмісту, а також широко використовується для активного блокування або пасивного виявлення цілого ряду нападів і зондувань.

Snort здатний виявляти:

− поганий трафік;

− використання експлойтів (виявлення Shellcode);

− сканування системи (порти, ОС, користувачі і т.д.);

− атаки на такі служби як Telnet, FTP, DNS, і т.д.;

− атаки DoS / DDoS;

− атаки пов'язані з Web серверами (cgi, php, frontpage, iss і т.д.);

− атаки на бази даних SQL, Oracle і т.д.;

− атаки по протоколам SNMP, NetBios, ICMP;

− атаки на SMTP, imap, pop2, pop3;

− різні Backdoors;

− віруси.

Snort можливо налаштувати для роботи в декількох різних режимах - режим аналізу пакетів, режим журналювання пакетів, режим виявлення мережевих вторгнень і вбудовуваним (inline) режим. Snort може бути налаштований для роботи в цих режимах:

Режим аналізу пакетів (Sniffer mode). Snort просто читає пакети які приходять з мережі і виводить їх на екран. В цьому режимі Snort діє просто як аналізатор, показуючи нефільтрований вміст середовища. Звичайно, якщо вам потрібно тільки аналізатор, можна застосувати Tcpdump або Ethereal, проте даний режим дозволяє переконатися, що все працює правильно і Snort бачить пакети.

Опції Snort показані в табиці 2.1.

Таблиця 2.1 – Опції Snort

Опція	Опис
-v	Виводить на екран IP і TCP / UDP / ICMP заголовки пакетів. (Verbose mode)
-d	Відображає дані прикладного рівня, використовується в докладному режимі виведення (опція -v) або в режимі журналювання пакетів.
-e	Включає висновок заголовків канального рівня.

Режим журналювання (протоколювання) пакетів (Packet Logger mode) дозволяє записувати пакети на диск для подальшого аналізу. Це корисно при проведенні аналізу за певний інтервал часу або перевірки змін в настройках і

політики безпеки. Щоб запустити Snort в режимі журналювання, потрібно скористатися тією ж командою, що і для режиму аналізу (-v, -d і / або -e), але з додаванням ключа –l каталог_журналів, що задає маршрутне ім'я каталогу журналів, в які Snort буде записувати пакети . Приклад:

snort -vde -l /var/log/snort

Ця команда створить файли журналів в каталозі / var / log / snort.

Режими виявлення мережевих вторгнень (Network Intrusion Detection System (NIDS) mode). Найбільш складний в конфігурації режим, який дозволяє аналізувати мережевий трафік і виконувати виявлення вторгнень на основі набору правил. В цьому режимі Snort протоколює підозрілі пакети.

Для перекладу Snort в режим виявлення вторгнень досить додати до наведеної вище інструкції ключ -c конфігураціонний_файл, який наказував би використовувати вказаний конфігураційний файл для управління протоколюванням пакетів. Конфігураційний файл визначає всі налаштування Snort, він дуже важливий.

Snort подається з імовірним конфігураційним файлом, але перед запуском в нього доцільно внести деякі зміни, що відображають специфіку вашого середовища.

Вбудовуваний режим (inline mode). Режим роботи спільно з файерволом iptables. Для того, щоб запустити в цьому режимі, необхідно додати додатковий ключ Q:/snort -GDc ../etc/drop.conf -l / var / log / snort. Перед запуском в цьому режимі необхідно переконатися, що програма встановлена з підтримкою даного режиму. Після цього слід налаштувати фаєрвол для взаємодії зі Snort.

Режими сигналізаціїSnort. При протоколюванні пакетів, викликають сигнали тривоги, необхідно вибрати підходящий рівень деталізації і формат "тривожних" даних.

В таблиці 2.2 перераховані опції, які можна задавати в командному рядку після ключа -A.

Таблиця 2.2 – Опції режиму сигналізації Snort

Опція	Опис
-A full	повна інформацію про сигнали, включаючи прикладні дані. Це якби уявляємо режим сигналізації. Він використовуватися за відсутності специфікаций
-A fast	швидкий режим. Протоколюються лише заголовки пакетів і тип сигналів. Це на вельми швидких мережах, якщо потрібно додаткова інформація, необхідно використовувати опцію full
-A unsock	відправляє сигнал в UNIX-сокетіз зазначеним номером, у якому може слухати інша программа
-A none	відключає сигнали тривоги

Ці опції слід конфігурувати під час компіляції за допомогою ключів інструкції configure. Приклад прописання створення файла куди заноситимуться ворожі чи підозрілі пакети. У файлі конфігурації створюється рядок на місці де прописується "Step6 :Output", і дописується рядок "outputalert_fast:alert.ids", зберігається і запускається як виявлення чи сигналізація, в папці snortlogalert.ids тепер записуються дані про підозрілі пакети. Можна ще вписати ключ "-Aconsole" тоді підозрілі пакети будуть дублюватися в консолі, як кажуть наочніше продемонстровані адміністратору.

Є також опції виводу (стандарт відправки повідомлень про події в системі) syslog, (формат повідомлень на основі протоколу спільного використання файлів Microsoft / 3Com, який використовується для передачі файлових запитів) smb і (база даних) database, але вони використовують не ключ -A, а окремі модулі виведення і пропонують більш широке розмаїття вихідних форматів. Ці опції слід конфігурувати під час компіляції за допомогою ключів інструкції configure.

SMB посилає сигнали тривоги службі спливаючих вікон Windows, тому можна побачити сигнали спливаючими на екрані, що здійснює моніторинг. Однак, перш ніж використовувати цю опцію, бажано ретельно налаштувати систему виявлення вторгнень, інакше не буде можливим щось робити, окрім як спостерігати спливаючі вікна. Для того щоб включити цей метод сигналізації, при

установці Snort задайте в інструкції configure опцію enable-smbalerts. Потім потрібно запустити snort з наступними аргументами:

snort -c /etc/snort.conf -M рабочі_станції

Задавши після -M імена хостів Windows, на які відправляються сигнали, Syslog посилає сигнали тривоги Syslog-сервера UNIX. Syslog - це служба, виконується на машині (зазвичай UNIX), яка може підхоплювати і зберігати різні файли журналів. Це допомагає консолідувати журнали вашої мережі в одному місці, а також ускладнює хакеру видалення протоколів вторгнень. Можна також визначити в конфігураційному файлі різні формати Syslog.

Snort безпосередньо підтримує чотири види виведення до бази даних з допомогою своїх модулів виведення. До підтримуваних форматів належать MySQL, PostgreSQL, Oracle iunixODBC. Це має задовольнити потреби більшості користувачів баз даних. І, природно, якщобаза даних не підтримується, можна розпочати проект написання потрібного модуля розширення. Модуль виведення до бази даних вимагає як параметрів часу компіляції, і настройок в конфігураційному файлі.

Архітектура Snort. Подивимося, з яких же функціональних блоків складається СОВ Snort. На Рисунку 2.5 показані компоненти, які включає в себе Snort.

Рисунок 2.5 – Функціональні блоки СОВ Snort

Основу Snort становить двигун, що складається з п'яти модулів.

Сніфер пакетів: даний модуль відповідає за захоплення переданих по мережі даних для подальшої їх передачі на декодер. Робить він це за допомогою бібліотеки DAQ (Data AcQuisition). Працювати даний сніфер може "в розрив" (inline), в пасивному режимі (passive) або читати мережеві дані з заздалегідь підготовленого файлу.

Декодер пакетів: даний модуль займається розбором заголовків захоплених пакетів, їх розбором, пошуком аномалій і відхилень від RFC, аналізом TCP-прапорів, винятком окремих протоколів з подальшого аналізу та іншої аналогічної роботи. Фокусується даний декодер на стеку TCP / IP.

Препроцесори: якщо декодер розбирав трафік на 2-му і 3-му рівні еталонної моделі, то препроцесори призначені для більш детального аналізу і нормалізації протоколів на 3-му, 4-му і 7-му рівнях. Серед найпопулярніших препроцесорів можна назвати frag3 (робота з фрагментованим трафіком), stream5 (реконструкція TCP-потоків), http_inspect_ (нормалізація HTTP-трафіку), DCE / RPC2, sfPortscan (застосовується для виявлення сканування портів) і різні декодери для протоколів Telnet, FTP , SMTP, SIP, SSL, SSH, IMAP і т.п. Деякі розробники пишуть свої препроцесори (наприклад, для промислових протоколів) і додають у власні системи виявлення вторгнень (IDS), побудовані на базі Snort.

Двигун виявлення атак. Даний двигун складається з двох частин. Конструктор правил збирає безліч різних вирішальних правил (сигнатур атак) в єдиний набір, оптимізований для подальшого застосування підсистемою інспекції захопленого і обробленого трафіку в пошуках тих чи інших порушень.

Модуль виведення. За фактом виявлення атаки Snort може видати (записати або відобразити) відповідне повідомлення в різних форматах - файл, syslog, ASCII, PCAP, Unified2 (двійковий формат для прискореної і полегшений обробки).

Розглянемо просте графічне представлення проходження даних через Snort. (рисунок 2.6).

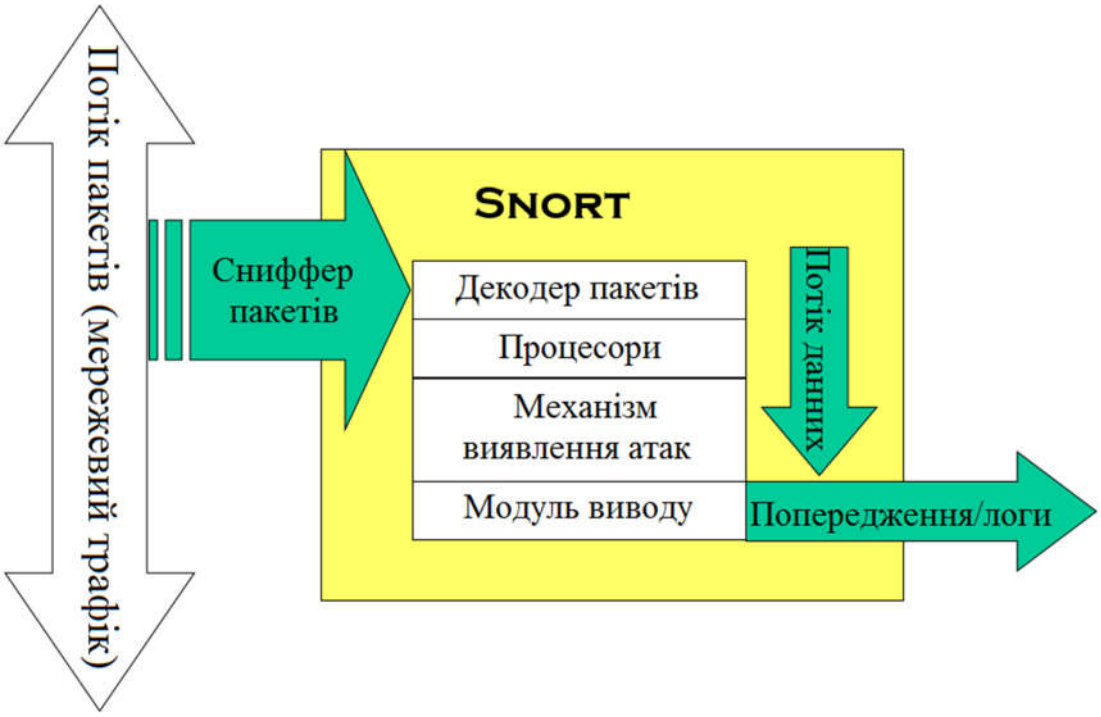

Рисунок 2.6 – Графічне представлення проходження даних через Snort

Snort and Wireless. Snort безпосередньо не вміє працювати з бездротовими мережами 802.11, але підключений до такого пристрою зможе інтерпретувати отриману інформацію. Сьогоднішній Snort в принципі не робить різниці в тому, з яким типом мережі він має справу, ніяких специфічних опцій при установці також задавати не треба. Як завжди, вказуємо інтерфейс '-i', і Snort починає аналіз пакетів в режимі raw monitoring (RFMON), без прив'язки до специфічної мережі, збираючи всі пакети, які трапляються в радіоефірі. Щоб контролювати тільки свою мережу, необхідно відповідним чином налаштувати Wi-Fi пристрій або систему фільтрів. Мережі IEEE 802.11 використовують три види пакетів: управління, контролю і даних.

Щоб відстежувати перші два типи, необхідно додати в рядок запуску параметр '-w'. Для більш ефективного захисту можна використовувати знаряддя нападу - Kismet який вміє відмінно сканувати ефір, а значить, його можна застосовувати для пошуку сторонніх пристроїв. Аналізуючи дані з різних точок доступу і Wi-Fi карт в зв'язці з Snort, можна дати відсіч навіть досвідченому хакеру.

Існує спеціальне відгалуження Snort - Snort-Wireless (snort-wireless.org), якраз призначене для виявлення атак, спрямованих на мережі стандарту 802.11. Snort-Wireless назад сумісний з Snort 2.0, при цьому містить деякі специфічні правила обробки пакетів, налаштованих на уразливості і типові атаки бездротових мереж. Функціонально Snort-Wireless мало відрізняється від Snort, додатково до його складу включено набір правил wifi.rules, що містить опис специфічних уразливостей і ряд препроцесорів. В даний час реалізовано п'ять препроцесорів:

1. antiStumbler (spp_antistumbler) - розпізнає розсилку великої кількості нульових SSID з одного MAC-адреси, що використовують NetStumbler і MacStumbler для виявлення точок доступу;

2. deauthFlood (spp_deauth_flood) - підраховує кількість кадрів деаутентіфікаціі і при перевищенні порога піднімає тривогу;

3. authFlood (spp_auth_flood) - визначає кількість спроб аутентифікації, що може свідчити про можливу DOS-атаки;

4. macSpoof (spp_macspoof) - відстежує спроби підміни MAC-адреси;

5. rogueAP (spp_rogue_ap) - його завдання повідомляти про присутність інших точок доступу.

У Snort таких препроцесорів немає, тому для захисту бездротової мережі має сенс використовувати саме Snort-Wireless.

2.3 Огляд утиліти Port Sentry

Головна мета PortSentry - виявити сканування портів на хості і відреагувати на такі перегляди.Це одна з причин, по якій PortSentry повинен розглядатися, як утиліта виявлення вторгнення. Якщо атакуючий не знає заздалегідь, які порти відкриті і доступні в системі, він почне сканування, для того щоб визначити доступні сервіси. У цей момент вступає активізується PortSentry. Інструмент контролює і UDP порти на системі і, в залежності від конфігурації, відповість на

ідентифікований перегляд. Мережева система безпеки IDS-Portsentry показана на рисунку 2.7.

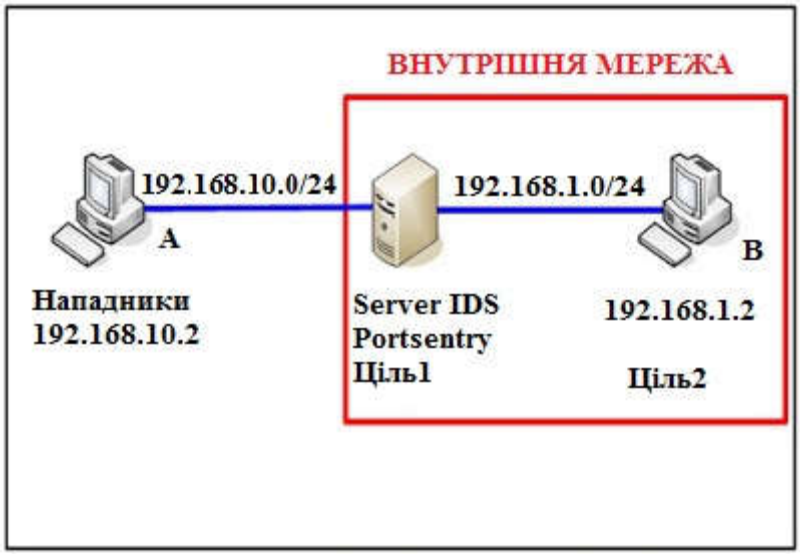

Рисунок 2.7 – Мережева система безпеки IDS-Portsentry

PortSentry контролює TCP і UDP перегляди і здатний виявивши більшість методів зірвати сканування інструментами, типу Nmap. Ось деякі з виявлених методів:

Connect scans - сканування з повним підключенням. У цьому методі використовується TCP функція connect () для встановлення з'єднання. У методі використовуються всі три стадії встановлення підключення, тому його легко виявити.

SYN scans - так само відомий як "half-open" scans - один з поширених методів прихованого сканування портів. У цьому методі використовуються тільки дві з трьох стадій встановлення з'єднання. Атакуюча система посилає TCP SYN пакет, як би запитуючи встановлення з'єднання. Хост відповідає їй SYN-ACK пакетом. Ініціатор потім посилає TCP RST пакет назад адресату, таким чином, закриваючи підключення. Ідея полягає в тому, що такі неповні підключення зазвичай не реєструються системою.

FIN scans - це сканування використовує пакети з встановленим прапором TCP FIN. Зазвичай FIN пакети використовуються тільки в заключній

послідовності підключення. Непередбачені FIN пакети, надіслані закритому TCPпорту, генерують неприпустимі пакети з іншого боку.

NULL SCANS - використовує пакет без встановлених TCP прапорців. Згідно RFC 793, це призведе до генерації неприпустимого пакета з іншого боку.

XMAS scans - використовує набір FIN, URG і PUSH TCP прапорців в TCP заголовку. Такий набір не може зустрітися в нормальному пакеті, тим самим, такий пакет змушує закритий порт згенерувати неприпустимий RST.

FULL-XMAS scan - цей тип сканування використовує всі встановлені TCP прапорці (SYN, ACK, RST, FIN, URG, PSH). Такі типи пакетів також в природі не зустрічаються.

UDP scan - цей тип сканування виявляється присутністю безлічі UDP пакетів, що виходять з окремого IP адреси.

Якщо PortSentry не може ідентифікувати перегляд, як один з вищезазначених, він використовує заданий за замовчуванням набір правил, які будуть ідентифікувати таке сканування. Більшість інших засобів виявлення сканування, на відміну від PortSentry, будуть фактично ігнорувати "непізнані" перегляди.

Одним з перешкод, з яким стикається PortSentry, це визначення, чи є пакет частиною сканування, або він становить частину нормального трафіку для даного порту. PortSentry використовує для цього два методи: спочатку він ігнорує порти, які знаходяться у використанні (пов'язані з відповідною службою). По-друге, користувач визначає список портів, контрольованих PortSentry, в файлі конфігурації. Якщо порт в списку файлу конфігурації збігається з використовуваним, то такий порт ігнорується PortSentry.

Одна з прекрасних особливостей PortSentry - це інтелектуальний контроль портів. Для проколів типу FTP, в яких клієнт відкриває порти в ефемерному діапазоні (TCP порти за 1024) і сервер встановлює зворотне підключення з хостом, PortSentry може досліджувати підключення які надходять, і якщо вони призначені для одного з ефемерних портів, воно буде проігноровано. Як тільки підключення завершено, PortSentry продовжить стежити за таким портом в звичайному режимі.

Перевірка переглядів TCP або UDP портів - не єдині методи, які використовуються для виявлення перегляду. Якщо довжина IP заголовка менше 5 (мінімальна довжина, зазначена в RFC 791), PortSentry реєструє таке повідомлення. Точно так же, якщо встановлені будь-які IP опції, PortSentry також звертає на них увагу. Хоча може здатися проблематично використання опцій типу source-routing, record-route або timestamp, вони можуть використовуватися нападаючим для збору розвідувальної інформації.

Внутрішня структура. У найпростішому випадку, PortSentry контролює порти на наявність потенційно підозрілих пакетів, реагуючи, якщо перегляд був виявлений. Щоб це здійснити, PortSentry повинен утримувати стан стеження за всіма IP адресами, які намагаються встановити підключення з хостом.

PortSentry використовує внутрішній движок стану в формі масиву IP адрес, який визначає, з'єднувався чи хост раніше з системою. Масив - двовимірна таблиця, що складається з IP-адреси і лічильника. Коли відбувається сканування адресата (випадковий або послідовний перегляд), PortSentry перевіряє масив, щоб визначити, чи був IP атакуючого помічений раніше, і, якщо це так, то лічильник збільшується. Цю процедуру він робить для кожного порту, контролюючи і виявляючи пакети, що приходять з IP-адреси атакуючого. Як тільки тригер досяг певного значення, PortSentry зреагує на перегляд.

Метод, яким PortSentry реагує на перегляд, змінюється, в залежності від налаштувань користувача в файлі конфігурації. PortSentry забезпечує три шляхи запобігання подальшого перегляду, які детально будуть розглянуті нижче. Усі три методи блокування хоста мають свої ризики і вигоди.

Методи блокування. Перший метод змінює таблицю маршрутизації так, що весь трафік з підозрілого IP буде посланий в певний "приймач". Використовуючи таблицю маршрутизації хоста, він буде переправлений за неправильною адресою від простежуваного хоста до неіснуючої IP адреси. В цьому випадку, PortSentry нагадує чорну діру - трафік приходить, але не повертається. Проблема з цим методом полягає в тому, що він збільшує розмір таблиці маршрутизації на хості. Оскільки таблиця маршрутизації зростає, необхідна більша кількість пам'яті для запам'ятовування маршрутів. Теоретично, нападник може підробляти вихідний

адресу пакетів, які використовуються в перегляді, змушуючи безладно PortSentry додавати записи в таблиці маршруту у відповіді. Це, теоретично, може привести до вичерпання системної пам'яті і створити проблеми в роботі інших додатків.

Другий метод - PortSentry може зв'язати різні пакети з програмами міжмережевого захисту (ipfw, ipfilter, ipfwadm, ipchains, і iptables) так, щоб правила міжмережевого захисту могли використовуватися для блокування трафіку з певного IP. PortSentry виявляє перегляд і додає відповідне правило до конфігурації міжмережевого захисту, яка далі заблокує IP адреса проглядається хоста. Знову, як і з використанням таблиць маршруту, для блокування трафіку, нападник може підробляти пакети, змушуючи PortSentry повідомляти міжмережевого захисту про блокування самих різних хостів без розбору. Це може привести до блокування законних хостів і доступних сервісів на сканованому хості.

Нарешті, правило TCP пакувальника для IP атакуючого може бути додано в системний /etc/hosts.deny файл. Він не блокує перегляд, а просто запобігає з'єднання від скануючого хоста до сервісів, захищених TCP пакувальником на простежуваному хості. Цей метод слабкий, в порівнянні з двома попередніми; проте, в цьому випадку важко викликати відмову в обслуговуванні проти скануючої системи.

Може здатися, що управління таблицями маршрутизації або використання правил міжмережевого захисту для блокування сканування, може легко використовуватися для блокування всього трафіку на сканованому хості, проте це не так. Так, є небезпеки у використанні цих методів; однак, належним чином конфігуруємо PortSentry і контролюючи його журнали реєстрації, цей ризик можна значно зменшити. Ось як автори PortSentry заявляють в документації:

"It is our experience though that spoofed scans are not an issue and we recommend people use auto-blocking knowing that% 99.9 of the time it will block a scan.

Again though, we strongly feel that the benefits of auto-blocking hosts * far outweighs * the limited risk you take by having auto-blocking turned on. "(Craig Rowland, Portsentry-2.0b1)"

Як тільки перегляд виявлений, PortSentry продовжує зберігати стан на попередньо блокованих хостах, записуючи IP адреса блокується хоста в файл реєстрації PortSentry в форматі <: protocol>: file (де - один з TCP або UDP). При виявленні перегляду PortSentry спочатку досліджує цей файл і тільки потім використовує власний внутрішній масив стану. Якщо хост був попередньо блокований, то PortSentry проігнорує таке сканування. Блокований файл кожного разу оновлюється при перезапуску PortSentry. PortSentry також записує блоковані хости в portsentry.history файл. Цей файл не буде перезаписано при перезапуску, і зберігає історії всіх блокованих хостів.

Компіляція PortSentry. Компіляція PortSentry проста. Потрібно завантажити gzipped tarball файл з Web сайту Psionic Technologies. Перед компіляцією програми, потрібно визначити кілька змінних у файлі portsentry_config.h:

– CONFIG_FILE - шлях до конфігураційним файлів PortSentry (за замовчуванням: /usr/local/psionic/portsentry2/portsentry.conf).

– WRAPPER_HOSTS_DENY - шлях і ім'я TCP wrappers hosts.deny файлу (за умовчанням: /etc/hosts.deny).

– SYSLOG_FACILITY – визначаємо чи використовує PortSentry засіб syslog (за замовчуванням: LOG_DAEMON).

– SYSLOG_LEVEL - Рівень syslog, який буде використовувати PortSentry (за замовчуванням: LOG_NOTICE).

Зазвичай змінна CONFIG_FILE повинна відповідати місцевої інсталяційною схемою на хості, і syslog_FACILITY повинен бути замінений на одне з LOG_LOCAL [0-7] значень. Зміна syslog_FACILITY до одного з значень LOG_LOCAL дозволяє адміністратору більш гнучко управляти файлом реєстрації. Інші дві змінні можуть бути залишені з заданими за замовчуванням значеннями.

Далі визначимо операційну систему, в якій буде використовуватися PortSentry. В даний час PortSentry визначає наступні системи: Linux, BSD, OpenBSD, FreeBSD, NetBSD, і generic. На Linux системі команда:

```
# Make linux
```

Скомпілюємо програму. Для того, щоб встановити програму:

Make install

За замовчуванням, програма буде встановлена в файл / usr / local / Psionic. Заданий за замовчуванням місце розташування може бути змінено в змінної INSTALLDIR в Makefile. Будь-які зміни цієї змінної повинні бути відображені в змінної CONFIG_FILE в файлі PortSentry_config.h, як зазначено вище. Після установки програми, буде потрібно її конфігурувати.

Конфігурація PortSentry. PortSentry управляється двома файлами:

1. portsentry.conf - основний файл конфігурації;

2. portsentry.ignore - список IP адрес, які PortSentry повинен виключити з спостереження.

Файл portsentry.ignore складається з простого списку IP адрес з відповідною маскою підмережі:

– 172.16.88.0/24;

– 10.16.17.0/24;

– 192.168.0.0/16;

– 127.0.0.1/32.

Цей файл повинен містити localhost (127.0.0.1) і IP адреси, призначені на інтерфейси місцевої системи.

У файлі portsentry.conf міститься інформація про те, як PortSentry повинен стежити і реагувати на перегляд портів. Перша частина файлу конфігурації визначає інтерфейс, який буде контролювати PortSentry, так як програма здатна контролювати тільки один інтерфейс одночасно. Для багатовузлових систем це не проблема. Однак на системах з двома або більше інтерфейсами, PortSentry буде слухати на первинному інтерфейсі. Одна очевидна проблема - контроль інтерфейсу модемного зв'язку. Portsentry не може звернутися до тип datalink, який використовується з P2P інтерфейсами модемного зв'язку, так що використання portsentry на портативних комп'ютерах в даний час не підтримується. Для

контролю тільки одного інтерфейсу на багатовузловій системі, такі змінні повинні бути встановлені у файлі portsentry.conf:

1. INTERFACE - встановіть в "AUTO" або до імені інтерфейсу, який повинен бути перевірений (наприклад, eth0, fxp0 і т.п.).

2. INTERFACE_ADDRESS - ця змінна повинна бути встановлена в IP адреса перевіреного інтерфейсу. Ця інформація не може бути автоматично визначена PortSentry.

Наступні два розділи конфігурують порти, які контролюватиме PortSentry, а також місця зберігання файлів історії, блокування та ігнорування. Psionic забезпечує деякі задані за замовчуванням списки портів, які можуть бути розділені на три групи: "comprehensive", "aware", і "bare-bones". Список "comprehensive" включає 54 TCP порту і 32 UDP порту в діапазоні від tcpmux до 54321 / TCP TCP (порт, іноді використовується троянами SchoolBus і BackOrifice 2000). Список "aware" складається з 30 TCP і 18 UDP портів з подібним розповсюдженням. Список "bare-bones" містить 24 TCP і 14 UDP портів.

Як згадано вище, portsentry.ignore файл - список IP адрес поряд з їх пов'язаними мережевими масками в 'slash' форматі, на які PortSentry не реагуватиме, якщо перегляд походить від одного з цих адрес. Цей список повинен бути коротким. Якщо вся локальна мережа включена в цей список, PortSentry ігноруватиме всі перегляди від локальних хостів, навіть якщо машина яку переглядали була скомпрометована атакуючим.

За замовчуванням, PortSentry не буде записувати IP адреси в ім'я хоста. Ця функція може бути включена, встановлюючи значення RESOLVE_HOST до 1 у файлі конфігурації, проте це не рекомендується, так як велика кількість підключень змусить PortSentry робити безліч DNS запитів, що може позначитися на продуктивності системи.

Наступний ряд розділів в файлі конфігурації визначає, як PortSentry повинен реагувати на виявлений перегляд. У цих розділах ви можете задати зміну маршруту, додавання хостів в TCP wrappers hosts.deny файл або запуск зовнішніх команд, як буде обговорено нижче. Змінні конфігурації, які фактично керують, чи відповідає portsentry на перегляд або не відповідає, це BLOCK_TCP і

BLOCK_DUP. Залежно від значень з цих двох змінних, portsentry може реєструвати перегляд (значення 0) але не відповідати, або відповідати на перегляд або блокувати його або виконувати зовнішні команди (значення 1 або 2 відповідно).

Блокування перегляду може бути зроблено декількома способами, два з яких взаємно виключені, в той час як третій може бути додаткових до будь-якого з перших двох. Щоб блокувати перегляд, PortSentry може або вставляти маршрут в таблицю маршрутизації хоста, яка направляє шкідливий трафік до "black hole", або може використовувати один з декількох фільтруючих програмних пакетів (ipfwadm, ipchains, ipf, ipfw, або iptables) щоб блокувати трафік від спостережуваного хоста. Дія визначається установкою змінної конфігурації KILL_ROUTE.

Змінюючи маршрут для підозрілого трафіку, PortSentry блокує подальші перегляди від атакуючого хоста. Зазвичай маршрут від такого хоста перенаправляється на бездіяльний IP або "dead host". Однак це метод може створити асинхронну маршрутизацію, за допомогою чого трафік входить в хост через один маршрут і виходить через інший. Також, цей тип захисту блокує тільки TCP перегляди, а UDP все ще можуть виконуватися наосліп.

Більш кращим методом блокування в PortSentry є використання пакетних фільтрів. Коли PortSentry вставляє додаткові маршрути в таблицю маршрутизації, при цьому збільшується розмір таблиці в OS і не забезпечується захист від UDP атак. Пакетні фільтри блокують тільки небажаний трафік і тому їх використання переважно.

Безладне блокування підозрілих IP адрес може призвести до відмови в обслуговуванні. Атакуючий може підмінити IP адреса, тим самим змушуючи систему блокувати (або змінювати маршрут) для таких адрес. Розглянемо два типових випадку.

1. Атакуючий, підміняє IP адреса DNS сервера, який використовує система. В цьому випадку portsentry зконфігурований з наступною KILL_ROUTE змінною:

KILL_ROUTE = "/ sbin / ipchains -I input -s $ TARGET $ -j DENY"

Перегляд вставить наступне правило фільтрації ipchains:

ipchains -I input -s -j DENY

І все пакети від DNS сервера будуть блоковані пакетним фільтром. Хост не зможе робити DNS запити. Будь-які сервіси, які вимагають виконання DNS запитів, не зможуть працювати.

2. У другому випадку сервіс на хості отримує запити від інших систем типу NNTP сервісу або NTP сервісу. Нападаючий може підробити адресу NNTP сервера, тим самим, блокуючи отримання Usenet новин. У разі блокування NTP сервера, може ускладнитися робота додатків, що вимагають точної синхронізації часу.

В обох випадках, наведених вище, PortSentry заблокував важливі хости, тому що атакуючий використовував підроблені перегляди. Хости, які використовують PortSentry, повинні також перевірятися (Psionic Technologies рекомендує використовувати програму LogSentry, яка контролює журнали реєстрації PortSentry), щоб гарантувати, що хост не заблокований.

Мінлива KILL_HOSTS_DENY забезпечує вставку IP скануючого хоста в etc / hosts.deny файл. Це забезпечує додатковий захист, якщо хост перезавантажений, і правило / маршрут пакетного фільтра, щоблокує трафік від переглядає хоста, був втрачений.

Ще одна опція в PortSentry - це можливість виконувати зовнішню команду, перед або після того, як хост був заблокований. Це конфігурується опцією KILL_RUN_CMD. Виконана ця команда, перш або після того, як PortSentry заблокує трафік від хоста джерела перегляду, визначається опцією KILL_RUN_CMD_FIRST. Установка цього параметра до '1', виконає команди перш, ніж PortSentry вживе заходів проти джерела перегляду. Значення "0" спершу заблокує хост і тільки потім виконає команду. Потрібно обережно використовувати цю опцію. Існує безліч засобів, які наприклад, атакуючий може використовувати nmap з опціями -s (для визначення адреси сканування) і -D <decoy1, [decoy2], ...> (для визначення додатково "приманки" адреси перегляду).

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ АЛГОРИТМУ ПРОЦЕСУ ВИЯВЛЕННЯ СКАНУВАННЯ ПОРТІВ

3.1 Початкове налаштування утиліти Snort

При установці Snort на операційну систему Windows можуть виникнути деякі складності. Тому в цій роботі приділяється досить докладна частина установки і можливостей настройки. Для початку потрібно скачати необхідні програми на робочий комп'ютер:

- winpcap;
- Snort;
- правила для Snort.

Все вищевказане скачується з офіційних сайтів цих додатків.

Winpcap - додаток, яке перехоплює і фільтрує пакети на рівні ядра. Це аналог вбудованому драйверу Unix систем libpcap. Установка не доставить особливих незручностей, запускається через звичайний інсталятор. Після цього потрібно завантажити з офіційного сайту саму IDS, після цього ми завантажуємо звідти ж свіжий архів з правилами. Наступним кроком стане повне копіювання всіх папок, які перебували в архіві з правилами в кореневій каталог додатки з повною заміною вмісту, де це потрібно. Потім для правильної роботи програми буде потрібно провести важливі зміни в файлі конфігурації:

var RULE_PATH c: \ snort \ rules
var SO_RULE_PATH c: \ snort \ so_rules
var PREPROC_RULE_PATH c: \ snort \ preproc_rules
dynamicpreprocessor directory c: \ snort \ lib \ snort_dynamicpreprocessor
dynamicengine c: \ snort \ lib \ snort_dynamicengine \ sf_engine.dll
#dynamicdetection directory / usr / local / lib / snort_dynamicrules

Знаходимо подібні рядки в файлі конфігурації і замінюємо тими які надані вище. Після цього пробуємо протестувати додаток. Запускаємо командний рядок і переходимо до каталогу додатка в розділ "bin". Введемо команду "snort -W". Результат роботи даної команди показано на рисунку 3.1.

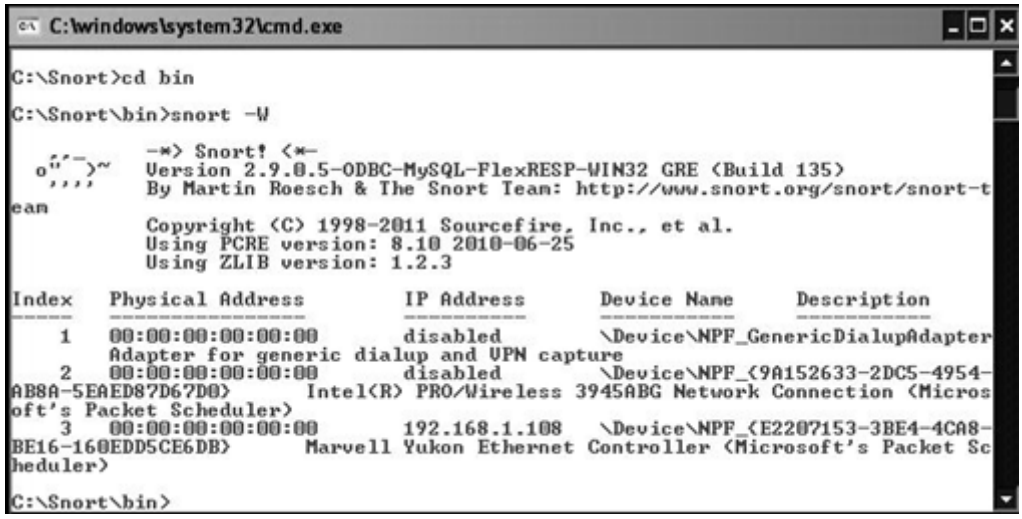

Рисунок 3.1 – Перевірка роботи

Цією командою ми перевіряємо працездатність додатки переглядати наші інтерфейси. Переконавшись що їх більше одного, вибираємо той який підключений до робочої мережі, щоб приступити до перехоплення пакетів і відстеження роботи IDS:

C: \ Snort \ bin \ snort -i 3 -c C: \ snort \ etc \ snort.conf -l C: \ snort \ log -A console

Розберемо тепер команду яку ми ввели. "- і 3" означає що ми будемо переглядати інтерфейс який має ID = 3 в списку наших інтерфейсів. Потім ми вказали шлях до файлу конфігурації і шлях до каталогу куди слід записувати "log" перехоплених пакетів. "-A console" позначає що тривожні пакети будуть виявлятися у нас в консолі. Якщо під час обробки виникають якісь неполадки усуваємо їх по ходу виявлення. Snort вказує рядок і вид помилки при складанні. Якщо все спрацювало, то ми нічого не побачимо до тих пір, поки не спрацює одне з запущених правил. Щоб задіяти одне з них спробуємо імітувати мережеву атаку

і запустимо підозрілий пакет з нашої локальної мережі що показано на рисунку 3.2. Для цього наприклад відкриємо командний рядок і введемо наступне: "Ping 192.168.1.16". Snort перехопить спробу прослухати хост під адресою 192.168.1.16 \ 24 і виведе повідомлення та інформацію про підозрілий дії в мережі. На жаль у подібних IDS систем є сильну нестачу це помилкові спрацьовування. У зв'язку з цим для того щоб Snort був корисним і не вводив в оману, потрібно досить ємко і чітко прописувати правила і розмежовувати популярні мережі, щоб уникнути цих помилкових спрацьовувань.

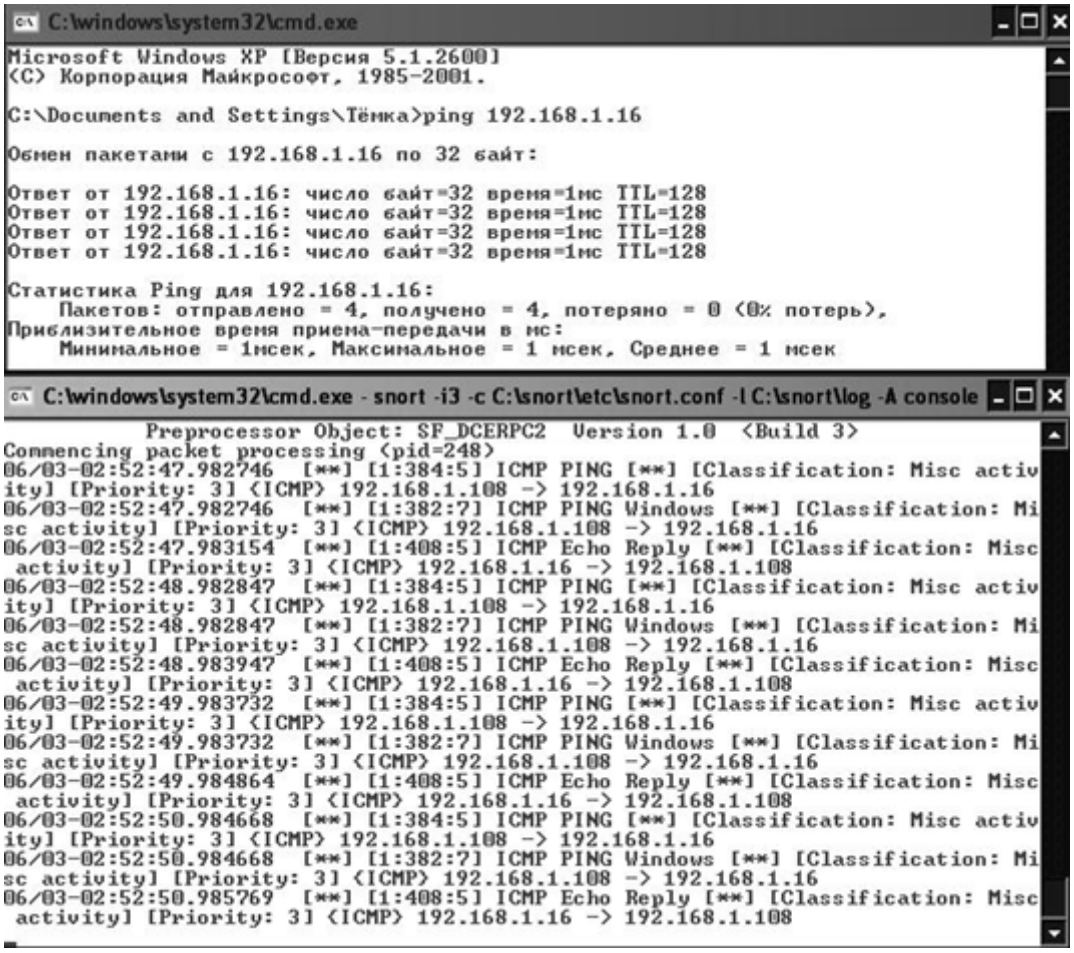

Рисунок 3.2 – Вивід Alert повідомлень

Зараз в консолі, де працює наш IDS, з'являться повідомлення про підозрілий пакет, який нагадує "прослуховування". Це задіяне правило показало, що Snort повністю працездатний. Розглянемо режими його роботи і синтаксис правил для подальшої роботи.

Режим аналізу пакетів. Режим який використовується скоріше для перевірки роботи ніж для фіксування атак. В цьому режимі Snort показує абсолютно все пакети і в залежності від опцій буде показувати або детальну інформацію або більш загальну. Обов'язковий ключ до написання є -v. Якщо він не буде вказано, то Snort за замовчуванням буде запускатися в інших режимах і видавати помилки в очікуванні інших ключів. У таблиці вказані можливі ключі режиму аналізу пакетів.

Таблиця 3.1 – Опції режиму аналізу пакетів

опція	Опис
-v	Видає на екран заголовки пакетів TCP / IP в мережі Ethernet
-d	Аналогічно попередній опції, але відображаються також дані прикладного рівня
-e	Аналогічно попередній опції, але видаються також заголовки канального рівня

Розглянемо приклад використання цього режиму на практиці і введемо команду:

"Snort -vde -i3", де -i3-номер інтерфейсу з якого слід зчитувати пакети. Snort починає швидко передавати на командний рядок безліч пакетів як показано на рисунку 3.3, щоб відключити режим натисніть "ctrl + x".

На малюнку зображена досить докладна інформація закладена в пакеті, а саме інформація канального і прикладного рівня.

Режим протоколювання пакетів. Режим подібний аналізатору. Єдина відмінна риса, це занесення інформації на вільне місце жорсткого діска.Snort протоколює інформацію про пакети, які були перехоплені і оброблені відповідно до вимог введених ключів адміністратора.

Щоб запустити Snort в режимі протоколювання, слід ввести аналогічні ключі, такі як в режимі аналізу (-v, d, e), і додатковий ключ –l (рисунок 3.3), задає маршрутне ім'я каталогу логів, в які Snort буде записувати пакети.

snort -vde -l / snort / log /

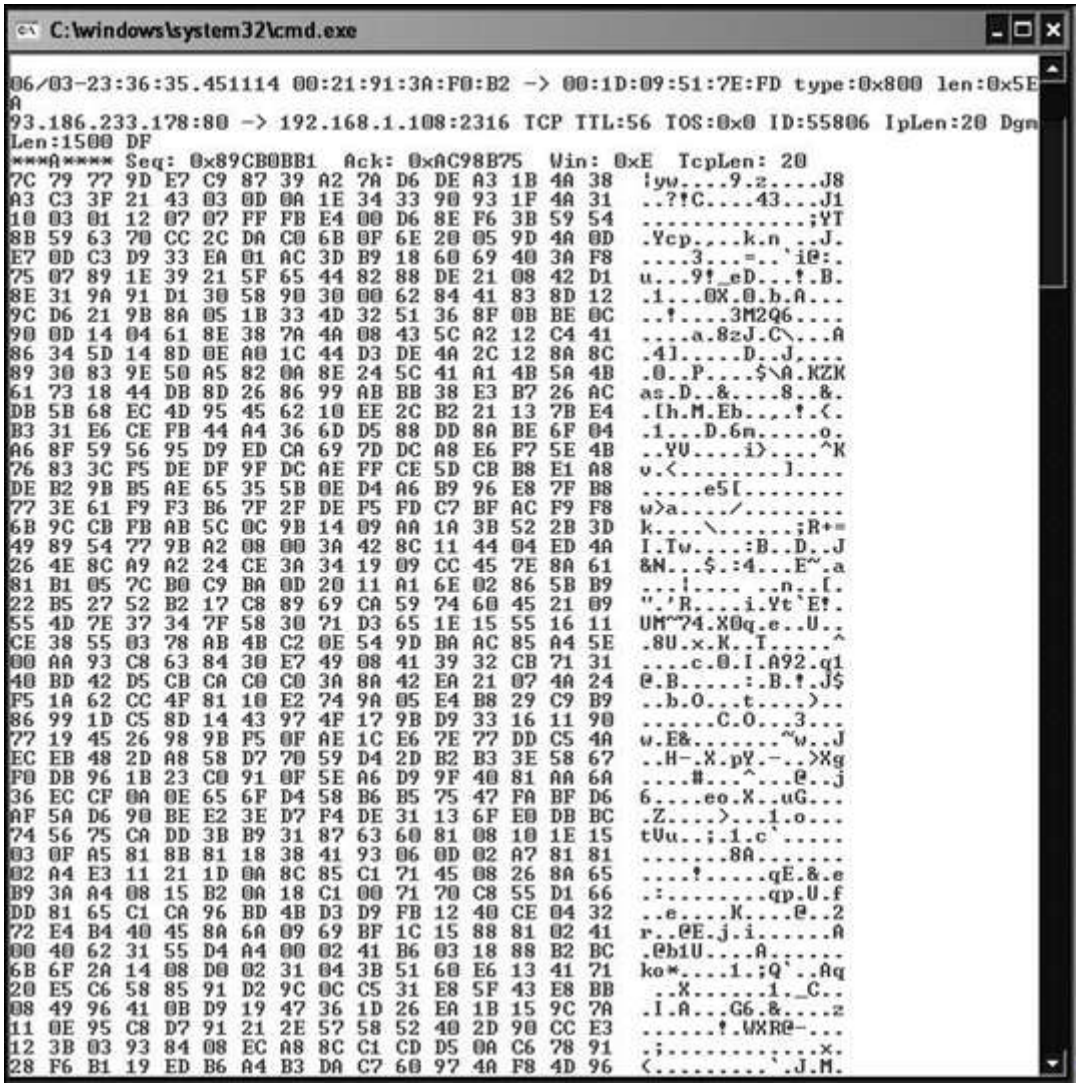

Рисунок 3.3 – Режим аналізу пакетів

Ця команда створить файли журналів в каталозі / snort / log /. Переконайтеся, що вказаний каталог існує, інакше програма не завантажуватися правильно. Так як додаток фіксує абсолютно все перехоплені пакети зі всіх можливих IP адрес, таке може відбуватися в мережі великої організації, можна звузити діапазон. Це робиться за допомогою команди -h HOME_NET, де HOME_NET - діапазон IP-адрес локальної мережі в нотації з косою рисою, за якою слідує префікс мережі. В цьому випадку Snort буде поміщати пакети в каталоги на основі нелокального IP-адреси в пакеті, що дозволяє легко розпізнавати "не місцевий" трафік. Якщо обидва хоста, цільової і вихідний, є локальними, Snort поміщає пакет в каталог, відповідний стороні з великим

номером порту, як би віддаючи перевагу підключати хосту перед серверним. У разі рівного розподілу номерів портів Snort за замовчуванням використовує вихідний адресу в якості каталогу для розміщення даних пакета. Зараз це може здатися несуттєвим, але якщо протоколювати сигнали про вторгнення, важливо швидко визначити, звідки виходить підозрілий трафік.

З огляду на наведені міркування, командному рядку для режиму протоколювання пакетів доцільно надати такий вигляд:

snort -vde -l / snort / log / -h 192.168.1.0/24

Тим самим внутрішня мережа задається діапазоном 192.168.1.1-254.

Можна також застосувати опцію -b для протоколювання всіх даних в одному бінарному файлі, придатному для подальшого читання за допомогою аналізатора пакетів. При протоколюванні з опцією -b немає необхідності визначати домашню мережу, так як дані будуть записуватися послідовно в один великий файл. Цей метод набагато швидше для протоколювання роботи активно використовуваних мереж або на повільних машинах. Він також полегшує аналіз за допомогою більш розвинених засобів, які доводиться застосовувати при перегляді великих обсягів перехоплених мережевих даних.

Режим виявлення вторгнень. Даний режим використовується для фіксування підозрілих і шкідливих пакетів, з подальшим занесенням в лог Snort'a. Відмітна особливість даного режиму полягає в додаванні ключа -з, який визначає шлях до конфігураційного файлу Snort. У цьому файлі міститься найважливіша і необхідна інформація, яка використовується для редагування роботи програми. Там вказуються адреси мережі, яку слід протоколювати і шукати вторгнення, що підключаються порти, дозволені файли правил шляху до баз даних куди можна буде заносити дані роботи Snort. Так само численні тонкі настройки конфігурації.

snort -de -l / snort / log -h 192.168.1.0/24 -c /snort/etc/snort.conf

Ми запустимо Snort в режимі виявлення вторгнень з використанням мається на увазі конфігураційного файлу snort.conf.

Режими сигналізації Snort. При протоколюванні пакетів, що викликають сигнали тривоги, необхідно вибрати відповідний рівень деталізації і формат "тривожних" даних.

Взаємодія з БД і Mysql. Snort безпосередньо підтримує чотири види виведення в базу даних за допомогою своїх модулів виводу. До числа підтримуваних форматів належать MySQL, PostgreSQL, Oracle і unixODBC. Це повинно задовольнити потреби більшості користувачів баз даних. І, природно, якщобаза даних не підтримується, можна взятися за проект з написання потрібного модуля розширення. Модуль виведення в базу даних вимагає як параметрів часу компіляції, так і налаштувань в файлі конфігурації.

Причини помилкових спрацьовувань і способи їх усунення формування правил. Типові причини помилкових спрацювань. Робота системи моніторингу мережі.

Багато організацій використовують системи моніторингу мережі, такі як HP OpenView або WhatsUp Gold, щоб стежити за системами в своїй мережі. Вони характеризуються високою мережевою активністю опитування і виявлення. Для опитування стану ці системи зазвичай застосовують SNMP або аналогічний протокол, але вони можуть також використовувати луна-тестування і інші, більш настирливі перевірки. За замовчуванням більшість систем виявлення вторгнень розглядають цю активність як шкідливу або, в крайньому випадку підозрілу. У великій мережі моніторинг може породжувати тисячі сигналів тривоги на годину, якщо система виявлення вторгнень налаштована для відстеження такої діяльності. Цього можна уникнути, ігноруючи активність за участю IP-адреси системи моніторингу. Можна також виключити з бази даних відповідні сигнали тривоги, якщо їх відстеження не представляє для вас особливої важливості.

Призначена для користувача активність. У різних організаціях, різні вимоги до обмежень працівників. За замовчуванням передача миттєвими повідомленнями, однорангове поділ файлів сигналізуватиметься як атака на мережу. Тому можна прибрати або навпаки озлобити подібну політику безпеки на

конкретні хости мережі організації, в залежності від вимог політики безпеки. Можна просто заносити в журнал повідомлення про подібну активності для того щоб дізнатися наскільки сильно ці дозволи впливають на смугу пропускання і на безпеку мережі в цілому.

Поведінка, що нагадує "троянську" програму або "хробака". Віруси і вірусоподібні програми і програмне забезпечення ("черв'яки", "троянські коні" і подібні програми), нерідко використовують мережеві інтерфейси і інші засоби, для отримання доступу і здійснення будь-яких шкідливих дій (отримання доступу до управління, зміни, видалення, копіювання інформацією атакується користувача). Подібну активність допомагають припинити IDS системи, проте подібні сигнатури можуть містити і звичайні нешкідливі програми, і тому слід відстежувати джерела сигнатур для більш точного встановлення шкідливим чи є пакет чи ні. Це допоможе уникнути надмірної кількості помилкових спрацьовувань.

Довгі базові ланцюжка аутентифікації. Подібні сигнали містити дуже великі довгі рядки входу WEB, деякі програми для виявлення вразливостей використовують такий метод для переповнення буфера і діставання несанкціонованого доступу. Однак зараз деякі сайти самі набивають в поле безліч інформації тим самим збиваючи столку IDS і змушуючи її фіксувати шкідливу сигнатуру.

Аутентифікаційна активність бази даних. Деякі мережеві системи виявлення вторгнень стежать за діяльністю з адміністрування баз даних. Теоретично в виробничих базах даних не повинно спостерігатися високою адміністративної активності, а її наявність може служити ознакою того, що хтось намагається зробити будь-які дії. Однак у багатьох базах даних використання йде паралельно з розробкою, звідси і великий обсяг адміністрування. Ця діяльність, хоча і цілком законна, буде породжувати безліч сигналів тривоги. Якщо ваша база даних знаходиться в стані безперервного розвитку, то вам, ймовірно, слід відключити ці сигнали, по крайней мере, поки база не стабілізується і не перейде в режим виробничої експлуатації.

Існує багато інших причин помилкових спрацьовувань, що залежать від конфігурації мережі і рівня активності. Якщо не приділяти достатньо уваги і не потрудитися організувати досить докладний і повний набір правил, то проблема помилкового спрацьовування зробить систему виявлення вторгнень якщо не марною, то дуже неефективною. Налаштування повинна відбуватися виходячи з знання вимоги безпеки в організації та мережі, обсягу мережі, доступу в інтернет і багатьох інших чинників які необхідно враховувати при цьому.

Приклади сигнатур мережевих систем виявлення вторгнень. Snort операційний мережевий вторгнення. Основний принцип по якому діють мережеві IDS це перевірка сигнатур і порівняння їх з відомими, якщо сталося збіг означає сигнатура визначається як підозріла. Як приклад можна розглянути атаку яка чітко буде ідентифікуватися IDS це атака cmd.exe спрямована проти Інформаційного сервера Інтернет -Web сервера корпорації Microsoft. Ця атака застосовується Інтернет - "хробаками" і вірусами, такими як Nimda і Code Red. Атакуючий "хробак" або людина намагається виконати в каталозі з правом на запис копію програми cmd.exe - командного інтерпретатора Windows, використовуючи переповнення буфера в модулі IIS, званому Internet Server API (ISAPI). У разі успіху хакер або черв'як отримує доступ до командного рядка на цій машині і може призвести значні руйнування. Однак команда для копіювання цього файлу є очевидною, і немає причини для її легального виконання користувачами через мережу за допомогою IIS. Тому, якщо ви бачите подібну активність, то досить імовірно, що це спроба вторгнення. Перевіряючи корисне навантаження пакета і розшукуючи слова cmd.exe, мережева система виявлення вторгнень може ідентифікувати дану атаку. На лістингу показаний один з таких пакетів. Шістнадцяткове подання вмісту знаходиться зліва, а переклад в текст - справа.

length = 55
000: 47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25 GET / scripts /..%
010: 35 63 25 35 63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 5c% 5c ../ winnt / sy
020: 73 74 65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F stem32 / cmd.exe? /

030: 63 2B 64 69 72 0D 0A c + dir ..

Синтаксис формування правил в Snort. Перед тим як почати писати правила для відстеження трафіку, слід навчитися правильно їх складати і ознайомитися з синтаксичними особливостями. Найкраще буде розібрати як формується правило на прикладі:

alert tcp any any → 192.168.0.0 / 24 80
(Content: "| 11 01 b6 a2 |"; msg: "example rule";)

Alert – генерувати сигнал з використанням обраного методу і записати інформацію про пакети в журнальний файл.

Tcp – протокол, який слід перевіряти на наявність умов. Далі по порядку вводиться IP адреса і порт відправника через стрілку адресу і порт одержувача, тобто кому направлений пакет. У дужках вказуються модулі які слід враховувати при фільтрації пакета. У нашому прикладі будуть позначатися пакети в яких буде знайдено збіг в 16-ковий коді "00 01 86 a5", і при збігу з умовами пакет буде позначатися сполученням "mount access".

Тема правила має вигляд:

<Дію><протокол><відправник><порт><напрямок><одержувач><порт>

Взагалі Snort має дуже багатий вибір правил, умов, і зарезервованих слів, які можна використовувати у формуванні правил, що дозволяє досить тонко відфільтрувати трафік в мережі.

Спробуємо написати правило, яке може відстежити відвідування користувачем будь-якої соціальної мережі, наприклад "vkontakte.ru". Для цього відкриємо будь-який з файлів папки \ snort \ rules \ і впишемо туди правило:

alert tcp any any → any any (msg: "ACCESS in Social web"; content: "vkontakte"; sid: 1;)

Розберемо це правило по командам. Це правило говорить що якщо в пакеті передається параметр "vkontakte", то пакет фіксується як тип alert і заноситься до відповідного файл звіту. Проглядається протокол tcp адреси мереж будь-які. У звіті такі пакети будуть позначатися сполученням "ACCESS in social web", що говорить нам про спробу доступу до соціальної мережі. Тепер адміністратор під час розгляду звітів може помітити звідки і коли відбувався вихід, на які не були погоджені політикою безпеки сайт. Так само слід зазначити наявність опції "sid1", вона потрібна для присвоєння правилом певного порядкового номера.Результати роботи показано у рисунку 3.4.

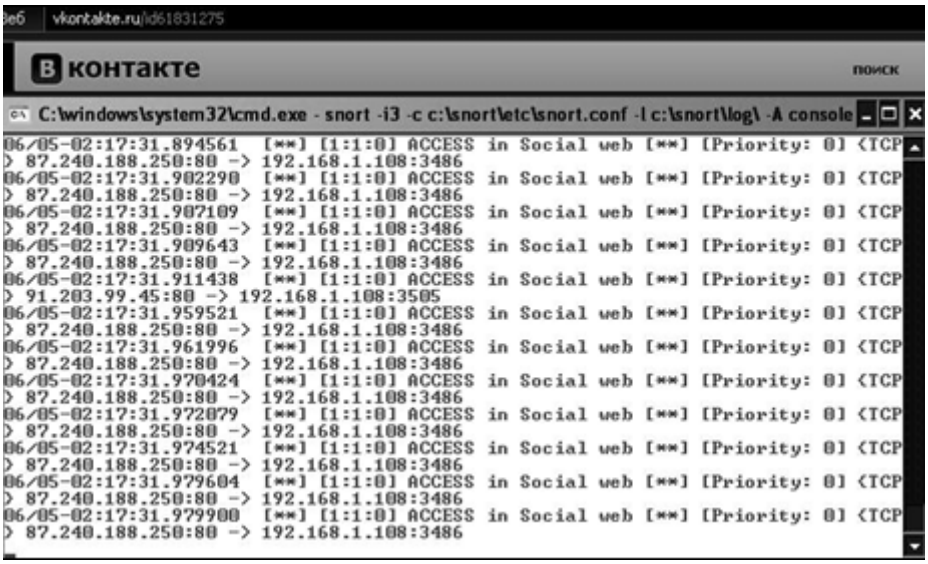

Рисунок 3.4 – Результат роботи правил

Правило не обов'язково в собі повинно містити контент будь-якого сайту або мережевого додатки. Існують цілі збірники інформації щодо написання правил, для цієї мережевої IDS, починаючи від можливості обмеження певних портів, закінчуючи перевіркою на наявність певних сегментів в пакеті.

Є один мінус докладного і чітко сформульованого правила, воно буде сильно навантажувати мережу і кожен пакет, відповідно для завантажених великих локальних мереж це може викликати деякі труднощі.

Написання plug-in для Snort. Для того щоб написати plug -in для Snort буде потрібно завантажити вихідний код з офіційного сайту www.snort.org, потім потрібно розпакувати отриманий пакет і потрапити в каталог \ templates, де

міститься чотири файли: sp_template.c, sp_template.h, spp_template. c, spp_template.h. Перші два файли використовуються для складання доповнення, а наступні два файли для препроцесорів. Після створення доповнення потрібно його інтегрувати в Snort, щоб він використовував його після компіляції. Інтеграція складається з трьох кроків:

1) Змінити plugbase.h і вставити рядок

#include <sp_template.h>

2) Змінити plugbase.c файл і в функції InitPlugins () додати назву функції після інших.

3) відкрити файл Makefile.am і додати імена файлів в список імен в рядок "snort_sources". Після цього виконуємо "Automake".

Аналогічні дії виробляємо для файлів препроцесорів. Якщо після зроблених змін Snort заробив і доповнення успішно запущено, отже, воно інтегровано.

3.2 Алгоритм виявлення процедури сканування портів

Snort готовий до запуску. Якщо в процесі запуску системи snort відбувається помилка про це повідомляє командний рядок з описом помилки і номером рядка конфігураційного файлу:

snort в режимі sniffer $ snort -dev
snort як демон $ snort -D

У разі, якщо snort виявляє збіг з правилом, то в файл / var / log / snort / alert дописується інформація про спрацювання того чи іншого правила.

Наприклад для того, щоб змусити snort писати не тільки заголовки, а й вміст пакета потрібно включити опцію "-d" для запису вмісту пакетів, або опцію "-b" для запису пакета цілком в бінарному вигляді.

Як вже зазначалося вище, IDS-системи, як і сам Snort, працюють на підставі списку правил, за якими вони і визначають чи є в що проходить трафіку "небезпечні" пакети чи ні. Так ось у ситеми Snort цей список правил виглядає таким чином.

Друга частина установки снорт - це модуль SnortSam. Цей модуль дозволяє пов'язувати систему Snort з місцевою Firewall-системою. Іншими словами це і є активна частина системи, яка дозволяє реагувати на небажаний трафік. Отже, установка модуля SnortSam. Є два варіанти: Перший (більш простий): - завантажуємо готові бінарники snortsam і вже модифікований snort з сайту http://www.snortsam.net/download.html - поміщаємо їх в каталог / usr / bin / Другий (для досвідчених користувачів) - завантажуємо архів з вихідними кодами snortsam - розархівовуємо - компілюємо snortsam (./configure, make, make install) - завантажуємо архів с патчем до snort - разархивируем - запускаємо pathsnort.sh як параметр вказуємо шлях до ісходникам snort: $ pathsnort.sh / root / snort-2.6.0.2 - компілюємо пропатченний snort (./configure, make, make install) Після встановлення модуля потрібно провести його налаштування. Поключаем плагін виведення для Snort (файл /etc/snort.conf):

img = snortsam-snortconf

192.168.1.253 - Хост на якому запущений файрволла і SnortSam. В якості експериментального фаєрвола ми вибираємо iptables.

Налаштовуємо SnortSam (файл /etc/snortsam.conf):

Accept <host> / <mask »,« key>

У цій опції перераховуються Snort сенсори, від яких SnortSam буде

Приймати пакети. Ви можете задати ім'я хоста, IP адреса, IP адреса і

Маску підмережі, а також ключ шифрування, іспользуемяй для цього хоста.

Або мережі.

Examples: accept 10.10.0.0/16, officepassword

Accept snort1, hostpassword

Accept 192.168.1.1

Якщо пароль відсутня, використовується ключ за замовчуванням DEFAULTKEY.

Ви можете задати тільки один хост в кожному рядку, але Ви можете використовувати

Необмежену кількість рядків.

Результат налаштування фаєрвола SnortSam командою Accept 192.168.1.253 показано на рисунку 3.5.

```
# accept <host>/<mask>,<key>
#
#   В этой опции перечисляются Snort сенсоры, от которых SnortSam будет
#   принимать пакеты. Вы можете задать имя хоста, IP адресс, IP адресс и
#   маску подсети, а так же ключ шифрования, используемяй для этого хоста
#   или сети.
#
#   Пример:    accept 10.10.0.0/16, officepassword
#              accept snort1, hostpassword
#              accept 192.168.1.1
#
#   Если пароль отсутствует, используется ключ по умолчанию DEFAULTKEY. Вы
#   можете задать только один хост в каждой строке, но Вы можете использовать
#   неограниченное количество строк.
#

accept 192.168.1.253
```

Рисунок 3.5 – Приклад правила accept

Дальше застосовуємо опцію bindip, яка заставляє Snortsam прослуховувати тільки один IP адрес або інтерфейс, замість всіх інтерфейсів або адресів.

bindip

#

За її використання Snortsam прослуховувати тільки ожин IP адреса (або

Інтерфейс), замість всіх інтерфейсів / адрес.

#

Приклад: bindip 192.168.0.1

Результат даної опції bindip 192.168.1.253, показано на рисунку 3.6.

```
# bindip
#
#   Эта опция заставляет Snortsam прослушивать только ожин IP адрес (или
#   интерфейс), вместо всех интерфейсов/адресов.
#
#   Пример:   bindip 192.168.0.1#
#

bindip 192.168.1.253
```

Рисунок 3.6 – Приклад правила bindip

Наступним кроком буде налаштування плагіну Iptables <adapter> <logoption>, який визиває Iptables для блокування хосту, шляхом додавання правила до набору активного списку блокіровки (рисунок 3.7). Ми повині вказати адаптер на якому здійснюється блокіровка, також ми можемо вказати опції логування.

Iptables <adapter> <logoption>

#

Цей плагін викликає iptables для блокування хоста, шляхом домавленія

Правила до набору активного списку блокування. Ви повинні вказати адаптер

На якому осущесвлять блокування (напр. Eth0), так само Ви можете вказати

Переконуємося, що незадані ніякі правила фільтрації для файрволла:

Iptables -L

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain FORWARD (policy ACCEPT)

target prot opt source destination

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

```
# iptables <adapter> <logoption>
#
#    Этот плагин вызывает iptables для блокирования хоста, путём домавления
#    правила к набору активного списка блокировки. Вы должны указать адаптер
#    на котором осущесвляется блокировка (напр. eth0), так же Вы можете указать
#    опции логирования.
#

iptables eth0
```

Рисунок 3.7 – Приклад логування

Рухаємося далі. IDS налаштована. Час для проведення експериментів. У натуральному виконанні нам знадобиться web-сервер (це трохи пізніше), поки ж проведемо емуляцію сервера за допомогою програми netcat.

Як вже було сказано вище, Snort працює з сигнатурами. Сигнатура - це сукупність байт індивідуально ідентифікують додаток. Наприклад будь WEB оглядач, перш ніж запросити HTML сторінку з сервера, посилає йому певну інформацію (представляється), для більш комфортного взаємодії (використання специфічних функцій працюють тільки в подавати додаток). Ця інформація, зокрема, включає в себе і унікальне ім'я (версію) броузера. Щоб це побачити, запустіть на своєму компьтере (в консолі) програму NetCat (яка дозволяє імулювать web-сервер), як показано нижче.

Nc -l -p 80

Прапори:

-l (listen) Запускає NetCat в режимі прослуховування порту;

-p (port) прослуховувати вказаний порт (в нашому випадку 80).

Всі оглядачі інтернет, за замовчуванням запитують WEB сторінку з 80 (http) порту. Тому ми буде емулювати WEB сервер, поки нам для цього вистачить тільки прослуховування порту сервера. Потрібно пам'ятати! Якщо ваш порт 80 зайнятий іншим додатком, вам необхідно закрити це додаток, щоб приклад працював правильно. Що це перевірити, потрібно запустити:

netstat -anb: для Windows

netstat -anp | grep 80: для Linux

Серед результатів необхідно знайти програму яка займає порт 80. Якщо такої немає, значить порт не зайнятий. Тепер потрібно запустити будь-який інтернет оглядач, і ввести адресу:

http: // localhost

Після цього NetCat повинен Вам показати все, що передаються оглядачем, HTTP заголовки. А броузер видасть помилку по якій він неможе знайти хост або

відкрити сторінку (так як NetCat не реалізує весь функціонал WEB сервера).

Нижче наведені приклади HTTP заголовків для InternetExplorer 6.0.2900 і для Firefox 2.0.0.2.

Ось як представився InternetExplorer:

- InternetExplorer ---------- --------

GET / HTTP / 1.1

Accept: image / gif, image / x-xbitmap, image / jpeg, image / pjpeg, application / x-shockwave-flash, * / *

Accept-Language: en-ca

Accept-Encoding: gzip, deflate

User-Agent: Mozilla / 4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Maxthon)

Host: localhost

Connection: Keep-Alive.

А Firefox представився таким чином:

- Firefox ---------- ----------

GET / HTTP / 1.1

Host: localhost

User-Agent: Mozilla / 5.0 (Windows; U; Windows NT 5.1; ru; rv: 1.8.1.2) Gecko / 20070219 Firefox / 2.0.0.2

Accept: text / xml, application / xml, application / xhtml + xml, text / html; q = 0.9, text / plain; q = 0.8, image / png, * / *;

q = 0.5

Accept-Language: ru-ru, ru; q = 0.8, en-us; q = 0.5, en; q = 0.3

Accept-Encoding: gzip, deflate

Accept-Charset: windows-1251, utf-8; q = 0.7, *; q = 0.7

Keep-Alive: 300

Connection: keep-alive

Серед різної службової інформації можна знайти рядки: MSIE і Firefox. За ним зазвичай і ідентифікується тип користувальницького оглядача.

Заголовки, ідентіфіцірующіі браузер, надсилаються сервера виключно як надлишкова інформація, ті. вони не передбачені специфікацією, як обов'язкові, отже їх може не бути зовсім. Так само ці заголовки еллементарно підробити (в Опері, наприклад, можна прямо в установках вказати тип браузера яким їй потрібно представлятися).

Припустимо Ви переконаний фанат відкритого вихідного коду і не хочете пускати комерційний InternetExplorer на свій WEB сайт. Реалізуємо цю ідею з допомогою Snort. Для початку навчимося просто розпізнавати наявність IE в Snort'e, а потім подивимося як можднo заблокувати доступ до сайту.

Припустимо у вас встановлений WEB сервер і Snort на одному комп'ютері. Щоб налаштувати Snort на виявлення InternetExplorer, додамо своє правило в стандартні правила Snort. За замовчуванням папка для правил розташований за адресою "/ etc / snortrules / rules". Для призначених для користувача правил існує спеціальні файл "local.rules". Спочатку правил там немає, додамо туди такий рядок:

alert tcp any any -> $ HOME_NET 80 (msg: "M $ InternetExplorer detected"; content: "MSIE"; nocase;)

Збережемо файл і перезапустимо Snort. Тепер зайдемо з будь-якого ком'ютера до сайт через IE. В логах Snort'a з'явитися відповідне повідомлення з нашого правила. Слід зауважити, що інші оглядачі (напр. FF) не викличуть ніякого логування.

Тепер навчимо Snort блокувати хост, з якого виробляється запит до нашого сайту через IE. Для цього знадобитися плагін для Snort'a Snortsam (який ми вже набудували).

Все що необхідно зробити це трохи відредагувати наше правило. Тепер воно повинно виглядати так:

alert tcp any any -> $ HOME_NET 80 (msg: "M $ InternetExplorer detected"; content: "MSIE"; nocase; fwsam: src, 1 hour;)

1 hour - (1 година) цей час на яке "зловмисник" буде заблокований. Спробуймо знову увійти на сайт за допомогою IE, і переконайтеся, що деякі сторінки сайту було повідомить вам ніякої інформації протягом години. А якщо не зміните броузер, то і ніколи.

Таким чином ми забраліковалі небажаний трафік. Зробили ми це з допомогою IDS Snort, її модуля SnortSam і фаєрвол IPtables. Потрібно розуміти, що це лише тривіальний приклад блокування небажаного трафіку. На прикладі браузера спрацювало. Тепер варто перевірити на прикладі спеціалізованих

мережевих додатків, наприклад мережевого сканера, що збирає інформацію по працюючому сервера. Отже, в якості сканера ми вибрали широко поширений і функціональний XSpider.

Поставимо задачу заблокувати мережевий сканер, на прикладі XSPider 7.5.

Для цього нам знадобиться встановлений WEB сервер. Для прикладу, виберемо широко відомий Apache. Спочатку нам необхідно знайти унікальну сигнатуру сканера.

Для наочності визначимо сигнатуру XSPider'a. Зробимо це використовуючи стандартні засоби логування сервера і за допомогою сніффер. Як сніффер будемо використовувати Wireshark, але підійдемо і будь-який інший. Все пріходяща від клієнтів запити Apche записує в файл access.log. У файлі конфігурації Apache (за замовчуванням httpd.conf), має бути присутня запис виду:

CustomLog logs / access.log common

Будемо використовувати цей факт для простеження дій XSPider (рисунок 3.8). Отже, запускаємо Wireshark на прослуховування локального інтерфейсу. А на другому хості запускаємо XPSpider. Додаємо хост для сканування на якому встановлений Web сервер, з контекстного меню хоста "Сканувати окремий сервіс" і вводимо порт 80 (порт web сервера). тепер натискаємо кнопку "Сканувати" і трохи почекаємо.

Перемістимо всю увагу на хост Web сервера. Переглянемо файл access.log (рисунок 3.9), перші запити сканера виглядають наступним чином:

192.168.1.111 - - [17 / Mar / 2018: Додати 12: 40: 11 +0300] "GET / HTTP / 1.1" 200 709

192.168.1.111 - - [17 / Mar / 2018: Додати 12: 40: 12 +0300] "GET SCANNER HTTP / 1.1" 400 342

192.168.1.111 - - [17 / Mar / 2018: Додати 12: 40: 12 +0300] "GET /hxaiktusqunkqknr.htm HTTP / 1.1" 400 395

192.168.1.111 - - [17 / Mar / 2018: Додати 12: 40: 13 +0300] "GET / HTTP / 1.1" 200 709

192.168.1.111 - - [17 / Mar / 2018: Додати 12: 40: 13 +0300] "GET / https-admserv / bin / index HTTP / 1.1" 404 297

192.168.1.111 - - [17 / Mar / 2018: Додати 12: 40: 13 +0300] "GET /Admin.po?proceed=yes HTTP / 1.1" 404 282

192.168.1.111 - - [17 / Mar / 2018: Додати 12: 40: 14 +0300] "GET /Admin/index.jsp HTTP / 1.1" 404 289

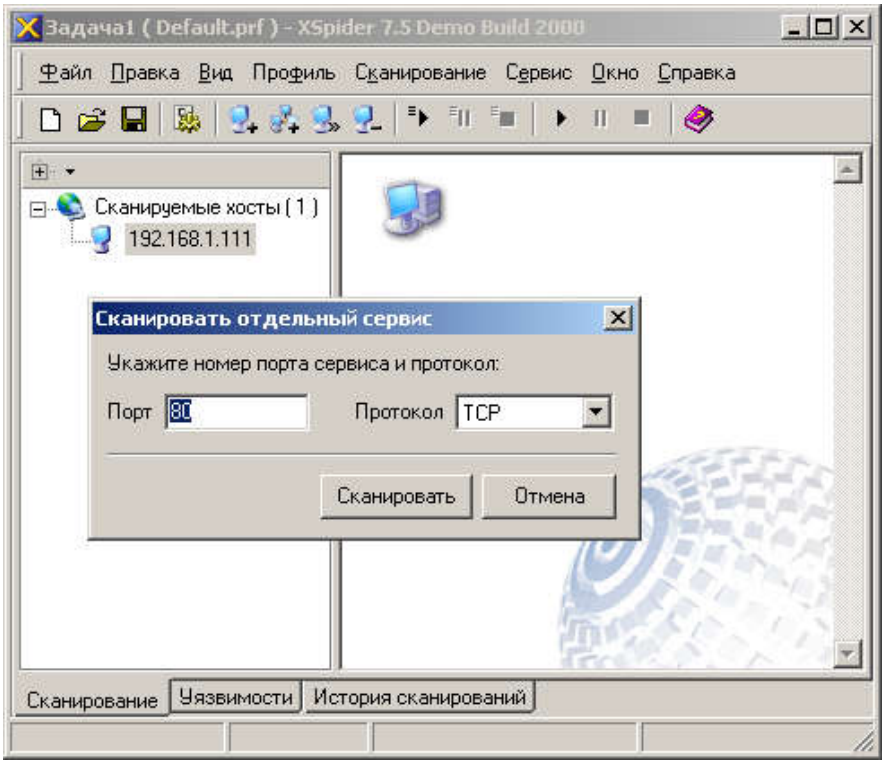

Рисунок 3.8 – Налаштування XSPider

Природно мало на якому сайті подібний файл буде існувати і ми можемо сприймати його як вітання XSPider'a. Серед результатів Wireshark'a теж можна побачити цю картину. Для більшої наглядносьті застосуємо фільтр:

(Ip.src == 192.168.1.105) && (http.request.method == "GET")

Запит "SCANNER" явно відрізняється від інших, хоча б тим, що не вказано в абсолютній формі, тобто не починається зі знака "/". багато сканери безпеки

спеціально залишають свої сигнатури при пошуках вразливостей. Тепер коли ми знаємо, як видається XSPider, ми можемо його заблокувати за допомогою системи Snort (за допомогою все того ж модуля Snortsam). Для цього додамо в файл "local.rules" рядок:

alert tcp any any -> $ HOME_NET 80 (msg: "XSpider detected"; content: "GET SCANNER HTTP / 1.1"; nocase; fwsam: src, 1 hour;)

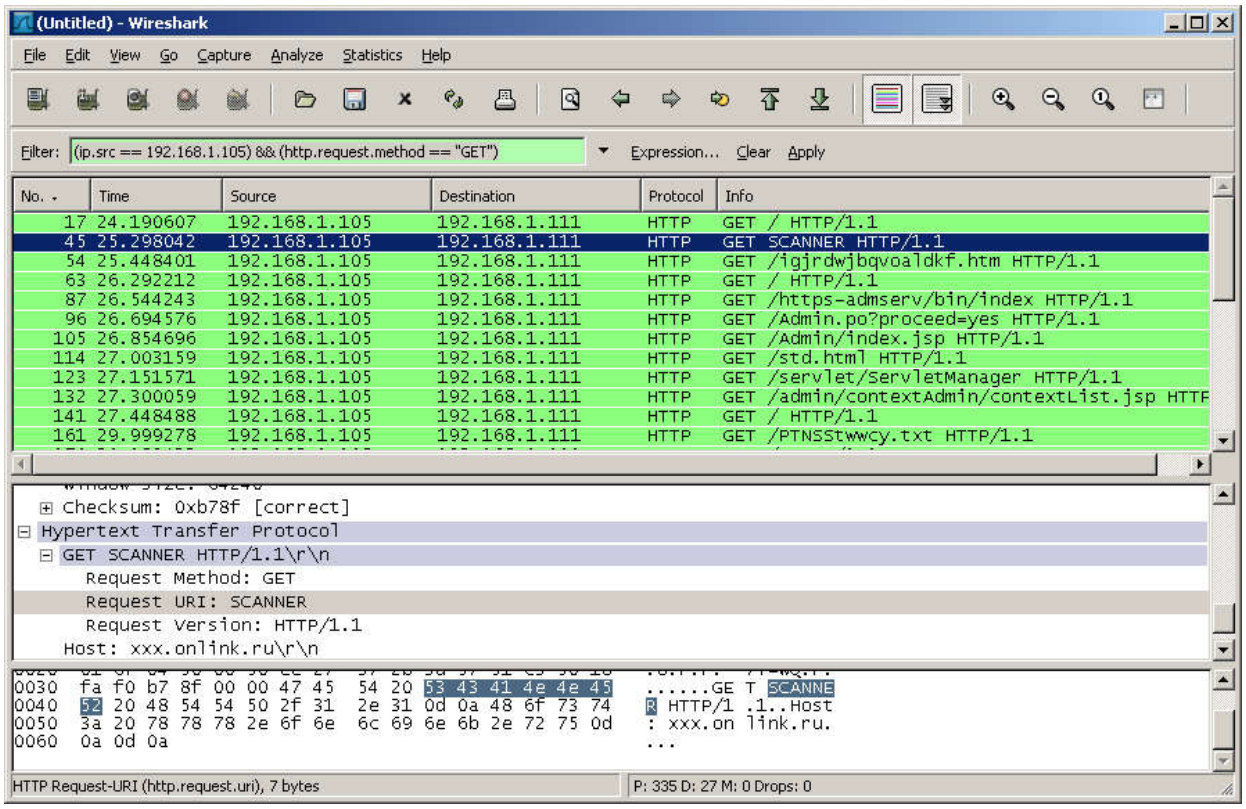

Рисунок 3.9 – Файл access.log

Після цього експерименту спробуємо просканувати XSPider'ом Web сервер ще раз. Результат: сканер заблокований на 1 годину.

Іншими словами ми заблокували спробу сканування з будь-якого IP адреси наш web-сервер з використанням сканера XSpider. Подібним чином можна налаштувати Snort на будь-який інший сканер (експлойт). Обов'язковою умовою, в даному випадку, має бути те, що ця утиліта повинна якось відрізнятися від звичайного трафіку (мати постійну сигнатуру, по якій її можна обчислити). Таким чином було розроблено блок схему роботи виявлення сканування портів в

корпоративній мережі (рисунок 3.10). В схемі показано як блокується Web-сервер на одну годину.

Запит "SCANNER" явно відрізняється від інших, хоча б тим, що не вказано в абсолютній формі, тобто не починається зі знака "/". багато сканери безпеки спеціально залишають свої сигнатури при пошуках вразливостей. Тепер коли ми знаємо, як видається XSPider, ми можемо його заблокувати за допомогою системи Snort (за допомогою все того ж модуля Snortsam). Для цього додамо в файл "local.rules" рядок:

alert tcp any any -> $ HOME_NET 80 (msg: "XSpider detected"; content: "GET SCANNER HTTP / 1.1"; nocase; fwsam: src, 1 hour;)

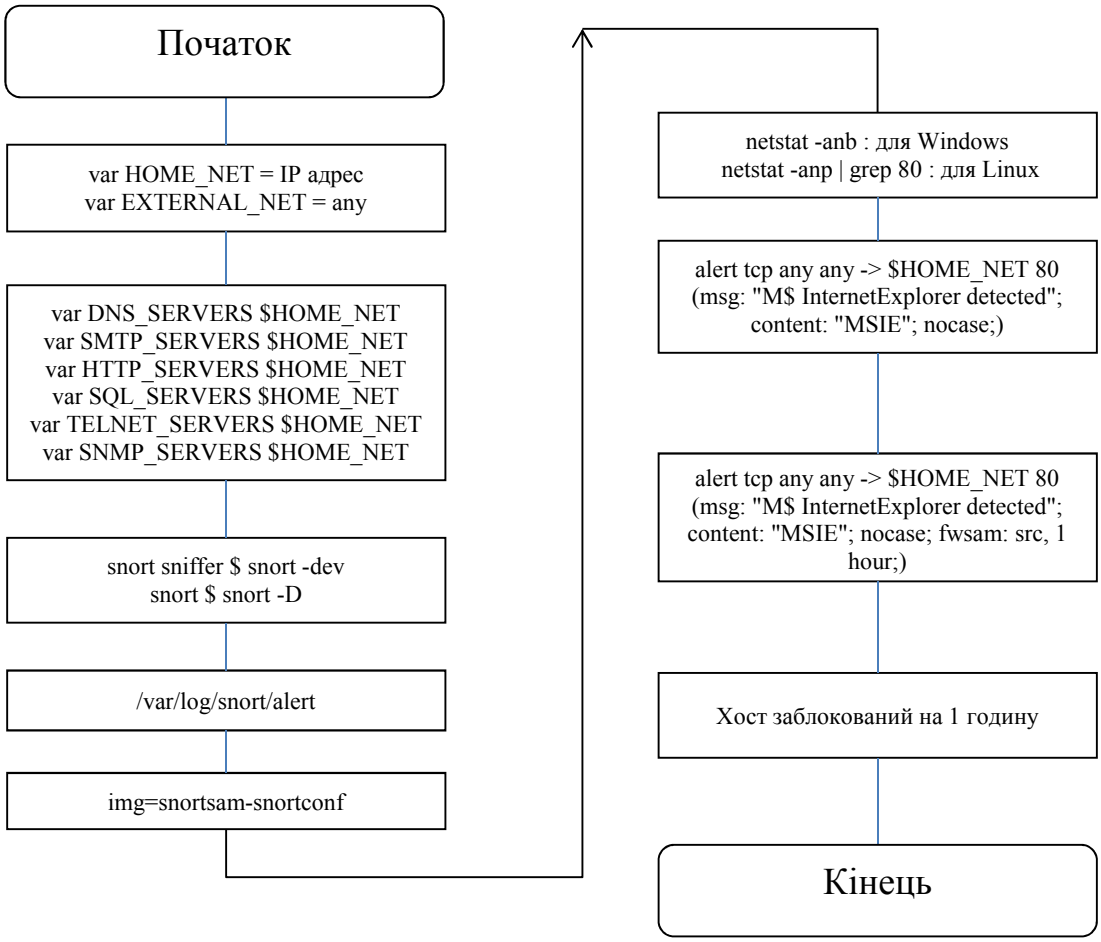

Рисунок 3.10 – Блок схема алгоритму процесу сканування портів

Після цього експерименту заради спробуємо просканувати XSPider'ом Web сервер ще раз. Результат: сканер заблокований на 1 годину.

Іншими словами ми заблокували спробу сканування з будь-якого IP адреси наш web-сервер з використанням сканера XSpider. Подібним чином можна налаштувати Snort на будь-який інший сканер (експлойт). Обов'язковою умовою, в даному випадку, має бути те, що ця утиліта повинна якось відрізнятися від звичайного трафіку (мати постійну сигнатуру, по якій її можна обчислити). var DNS_SERVERS HOME_NET

3.3 Додаткові утиліти для системи Snort

Analysis Console for Intrusion Databases (ACID) - аналітична система, що надає Web-інтерфейс (використовується PHP) для перегляду результатів аналізу лог-файлів брандмауерів, NIDS, мережевих діагностуючих програм (рисунок 3.10). Версія 0.9.6. працює в середовищі всіх ОС, що підтримують PHP (Linux, * BSD, Windows, Solaris). Ліцензія - GNU GPL.

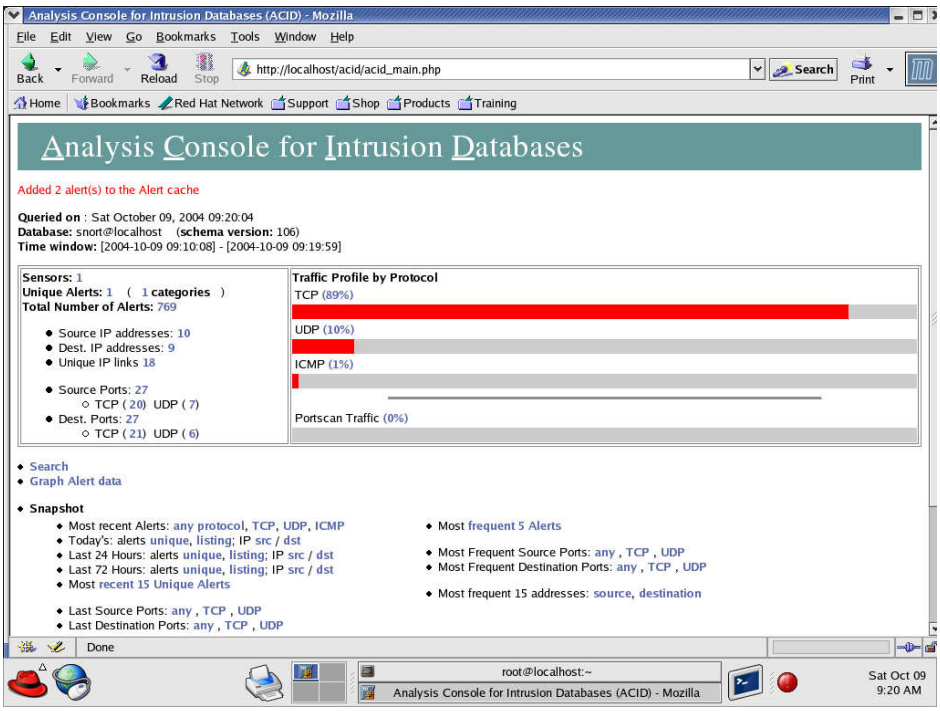

Рисунок 3.10 – Інтерфейс програми ACID

Intrusion Detection Exchange Architecture (IDEA) - система розподіленого контролю безпеки мережі. Поточна версія - 1.0.2. Використовуючи архітектуру клієнт-сервер, вона пов'язує безліч NIDS-систем, об'єднуючи їх дані в інформаційний базис і забезпечуючи аналіз в режимі реального часу. IDEA реалізована як системо незалежна програма, яка застосовує технологію Javam (рисунок 3.11).

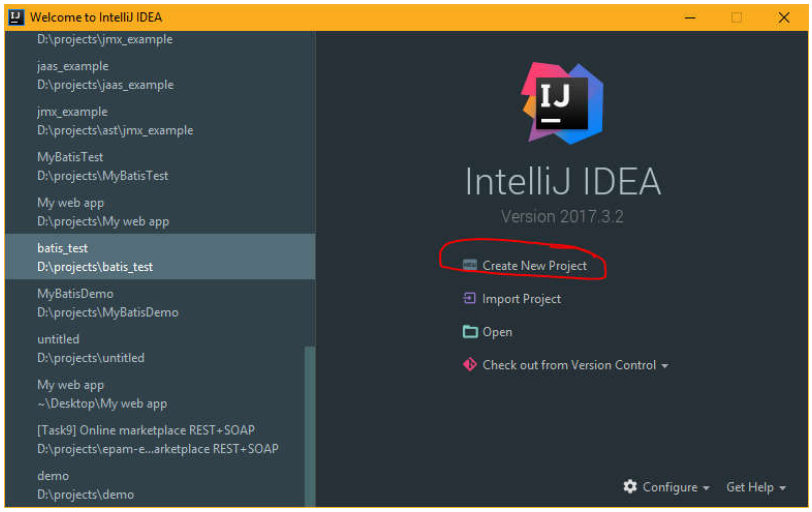

Рисунок 3.11 – Інтерфейс програми IDEA

SnortConf - графічна оболонка для Snort. Розроблена на основі бібліотеки Curses, програма SnortConf надає простий і інтуїтивно зрозумілий інтерфейс - меню з адміністрування Snort. Версія - 0.4.2-1 (рисунок 3.13). Призначена для POSIX-сумісних систем (Solaris, * BSD, Linux і т.д.).

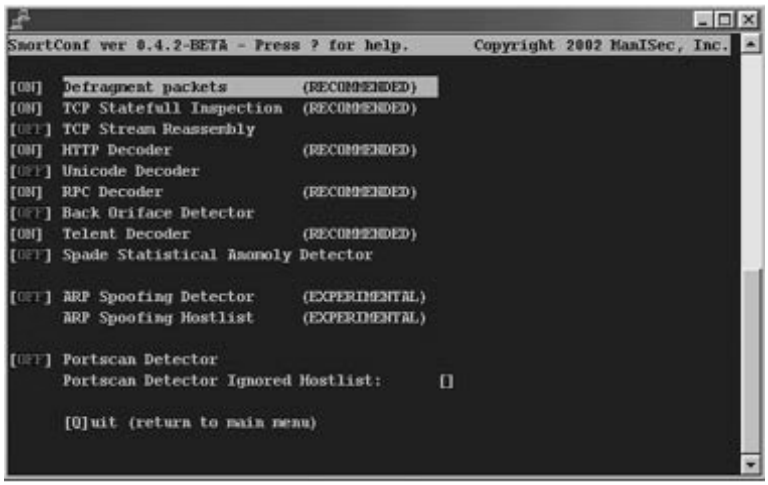

Рисунок 3.13 – Інтерфейс програми SnortConf

IDScenter - графічна оболонка Snort-Win32 (рекомендується використовувати Windows 7 / 8 / 10) (рисунок 3.14). Поточна версія - 1.1. IDScenter значно полегшує завдання управління, контролю і моніторингу Snort IDS. Включає функції діагностики конфігурації, підтримує сигнали тривоги Snort. При визначенні атаки можна запустити зовнішню програму.

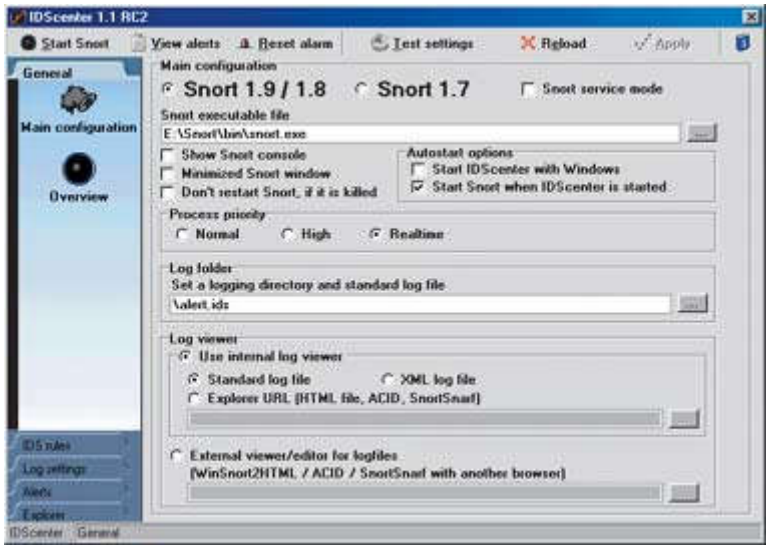

Рисунок 3.14 – Інтерфейс програми IDScenter

Intrusion Detection Exchange Architecture (IDEA) - система розподіленого контролю безпеки мережі. Поточна версія - 1.0.2 (рисунок 3.12). Використовуючи архітектуру клієнт-сервер, вона пов'язує безліч NIDS-систем, об'єднуючи їх дані в інформаційний базис і забезпечуючи аналіз в режимі реального часу. IDEA реалізована як системо незалежна програма, яка застосовує технологію Java.

ВИСНОВКИ

В роботі розв’язано актуальну на сьогоднішній день задачу по дослідженню та удосконаленню алгоритму виявлення процесу сканування портів в корпоративній комп’ютерній мережі, що забезпечиють ефективне виявлення і блокування мережевих атак в комп’ютерних мережах та системах. При цьому отримано наступні результати.

1. Розкрито теоретичні основи сканування портів в комп’ютерній мережі, зокрема, методи сканування за допомогою Nmap, типи сканування: SYN-сканування, TCP-сканування,UDP-сканування,ACK-сканування,FIN-сканування. Результати проведеної роботи показало нам переваги та недоліки кожного методу та типу сканування а також принцип роботи даного методу атаки на мережу.

2. Показано доступність використання утиліт сканування портів в компютерній мережі що ще раз підтвердило необхідність захисту мережі від зовнішніх атак.

3. Досліджено алгоритм процесу виявлення та блокування сканування портів на основі утиліти Snort, яка забезпечує виявлення та блокування мережевих атак, і записує всі дані трафіку в базу даних, а також системи PSAD і утиліти PortSentry. Враховуючи низьку складність реалізації алгоритму дана утиліта планується використати для підвищення захисту мережі, передачі даних та використання ресурсів Інтернету. Дослідження даних алгоритмів дало змогу нам визначити найбільш підходящу утиліту для роботи.

4. Обгрунтовано вибір утиліти та додаткових допоміжних програм для забезпечення надійної роботи алгоритму виявлення процесу сканування портів.

5. Розроблено унікальні правила для конкретної корпоративної компютерної мережі що використовуються в утиліті Snort це дало змогу забезпечити більш надійний захист мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Довгий В.В., Алгоритми сканування портів у корпоративній комп'ютерній мережі. Актуальні задачі сучасних технологій: семінар комп'ютерні науки та інформаційні технології. / Небесний І.В. // Тернопіль: ТНЕУ, 2018. – С. 27.

2. Smetters, D.K., Talking to strangers: authentication in ad-hoc wireless networks. / Balfanz, D., Stewart, P., Wong, H.C. // In: NDSS 2002 – p. 103–104.

3. Bonomi, F., Fog computing and its role in the internet of things / Milito, R., Zhu, J., // In: Workshop on Mobile Cloud Computing. ACM 2012 – p. 84–85

4. Bouzefrane, S., Cloudlets authentication in nfc-based mobile computing. / Mostefa, A.F.B., Houacine, F., Cagnon, H.,//In: MobileCloud. IEEE, 2014 – p. 33–38.

5. Cao, N., Privacy-preserving multi-keyword ranked search over encrypted cloud data. / Wang, C., Li, M., Ren, K., Lou, W., // TPDS 25(1), 2014 – p. 222–233.

6. Cao, N., Lt codes-based secure and reliable cloud storage service. / Yu, S., Yang, Z., Lou, W., Hou, Y.T // In: INFOCOM. IEEE,.2012 – p. 54–66.

7. Cash, D., Dynamic searchable encryption in very-large databases: data structures and implementation. // In: NDSS, et al. vol. 14, 2014 – p. 145–151.

8. Damiani, E., A reputation-based approach for choosing reliable resources in peer-to-peer networks. // In: CCS. ACM, et al. 2002 – p. 123–144

9. Dinh, H.T., A survey of mobile cloud computing: architecture, applications, and approaches. / Lee, C., Niyato, D., // WCMC 13(18), 2013 – p. 1587–1611.

10. Dsouza C., Policy-driven security management for computing: preliminary framework and a case study. / Taguinod, M., //: IRI. IEEE, 2014 – p. 51–56.

11. Tilborg, H.C.A., Differential privacy. / Jajodia, S. // In: Encyclopedia of Cryptography and Security. LNCS, vol. 2011. Springer, Heidelberg Dwork, C., 2011 – p. 344–367.

12. Gao, Z., Location privacy in database-driven cognitive radio networks: Attacks and countermeasures / Zhu, H., Liu, Y., Li, M., Cao, Z., // In: INFOCOM. IEEE, 2013 – p. 6–10.

13. Gennaro, R., Non-interactive verifiable computing: outsourcing computation to untrusted workers. / Gentry, C., Parno, B., In: Rabin, T. // CRYPTO 2010. LNCS, vol. 6223. Springer, Heidelberg, 2010 – p. 465–482.

14. Jajodia, S. Top10 technology predictions, Gil Press // Idc, 2015 – p. 3-4..

15. Ha, K., Towards wearable cognitive assistance / Chen, Z., Hu, W., Richter, W., Pillai, P., Satyanarayanan, M., // In: Mobisys. ACM, 2014 – p. 45–49.

16. Han, H., A measurement based rogue ap detection scheme. / Sheng, B., Tan, C.C., Li, Q., Lu, S., // In: INFOCOM. IEEE, 2009 – p. 138–165.

17. Han, H., A timing-based scheme for rogue ap detection, / Sheng, B., Tan, C.C., Li, Q., Lu, S., // TPDS 22(11), 2011 – p. 1912–1925.

18. Jøsang, A., A survey of trust and reputation systems for online service provision. / Ismail, R., Boyd, C., // Decis. Support Syst. 43(2), 2007 – p. 618–644.

19. Klaedtke, F., Access control for sdn controllers. / Karame, G.O., Bifulco, R., Cui, H, // In: HotSDN, vol. 14, 2014 – p. 1140–1192.

20. .Lu, R., Eppa: an efficient and privacy-preserving aggregation scheme for secure smart grid communicationset al, // TPDS 23(9), 2012 – p. 1621–1631.

21. McKeown, N. Openflow: enabling innovation in campus networks, // ACM SIGCOMM CCR 38(2), 2008 – p. 69–74.

22. McLaughlin, S., Protecting consumer privacy from electric load monitoring. / McDaniel, P., Aiello, W., // In: CCS. ACM, 2011 – p. 120–133.

23. J. Netw..A survey of intrusion detection techniques in cloud. // Comput. Modi, C., et al., Appl. 36(1), 2013 – p. 42–57.

24. Novak, E., Near-pri: Private, proximity based location sharing. / Li, Q., // In: INFOCOM. IEEE, 2014 – 45–53.

25. Parno, B., Nearly practical verifiable computation. / Howell, J., Gentry, C., Raykova, M.: Pinocchio, // In: Security and Privacy. IEEE, 2013 – p. 456–487.

26. Qin, Z., Defending against unidentifiable attacks in electric power grids. / Li, Q., Chuah, M.C., // TPDS 24(10), 2013 – p. 1961–1971.

27. Qin, Z., Preserving secondary users' privacy in cognitive radio networks. / Li, Q., Zamkov, D,//In: INFOCOM, 2014 Proceedings IEEE. IEEE, 2014 – p. 66.

28. Rial, A., Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society. ACM, / Danezis, G. // Privacy-preserving smart metering. In, 2011 – p. 167–169.

29. Satyanarayanan, M., The case for vm-based cloudlets in mobile computing. / Bahl, P., Caceres, R., Davies, N., // Perv. Comput. 8(4), 2009 – p. 14–23.

30. Satyanarayanan, M., An open ecosystem for mobile-cloud convergence. // et al. Commun. Mag. 53(3), 2015 – p. 63–70.

31. Shi, Y., Cloudlet mesh for securing mobile clouds from intrusions and network attacks. / Abhilash, S., Hwang, K., // In: Mobile Cloud 2015 – p. 543–556.

32. Shin, S., network security monitoring using openflow in dynamic cloud networks. / Gu, G.: Cloudwatcher, // In: ICNP. IEEE, 2012 – p. 67–86.

33. . Song, D.X., Practical techniques for searches on encrypted data / Wagner, D., Perrig, A., // In: Security and Privacy. IEEE, 2000 – p. 44–57.

34. Stojmenovic, I., The fog computing paradigm: scenarios and security issues. / Wen, S., // In: FedCSIS. IEEE, 2014 – p. 684–698.

35. Takabi, H., Security and privacy challenges in cloud computing environments. / Joshi, J.B., Ahn, G.J., // IEEE Secur. Priv. 8(6), 2010 – p. 24–31.

36. Tsugawa, M., Cloud computing security: what changes with software-defined networking? / Jajodia, S., Kant, K., Samarati, P., Singhal, A., Swarup, V., Wang, C. // Secure Cloud Computing. LNCS. Springer, Heidelberg, 2014 – p. 90.

37. Valenzuela, J., Real-time intrusion detection in power system operations. / Wang, J., Bissinger, N., // IEEE Trans. Pow. Syst. 28(2), 2013 – p. 1052–1062.

38. Vaquero, L.M., Finding your way in the fog: towards a comprehensive definition of fog computing. / Rodero-Merino, L., // ACM SIGCOMM CCR 44(5), 2014 – p. 27–32.

39. Wang, C., Enabling secure and efficient ranked keyword search over outsourced cloud data / Cao, N., Ren, K., // TPDS 23(8), 2012 – p. 1467–1479.

40. Wang, C., Privacy-preserving public auditing for data storage security in cloud computing / Wang, Q., Ren, K., Lou, W., // In: INFOCOM. IEEE, 2010 – p. 47.

41. Wei, W., Flexible privacy-preserving location sharing in mobile online social networks. / Xu, F., Li, Q.: Mobishare, // In: INFOCOM. IEEE, 2012 – p. 142.

42. Willis, D.F., Paradrop: a multi-tenant platform for dynamically installed third party services on home gateways. / Dasgupta, A., Banerjee, S., // In: SIGCOMM Workshop on Distributed Cloud Computing. ACM, 2014 – p. 376–389.

43. Yang, K., Data storage auditing service in cloud computing: challenges, methods and opportunities. / Jia, X., // World Wide Web 15(4), 2012 – p. 409–428.

44.Обнаружение сетевых атак – Snort [Електроний ресурс]. – Режим доступу:URL: http://www.compdoc.ru/network/local/snort/ (дата звернення 25.05.18).

45.Энциклопедия сетевых протоколов [Електроний ресурс]. – Режим доступа: URL: http://www.protocols.ru/ (дата звернення 15.04.18).

46.Атаки на сеть [Електроний ресурс]. – Режим доступу: URL: http://big-bro.info/ataki-na-set/ (дата звернення 22.05.18).

47.Русская группа Snort [Електроний ресурс]. – Режим доступу: URL: http://www.snortgroup.ru/node/25 (дата звернення 27.05.18).

48.Ищейка по имени Snort [Електроний ресурс]. – Режим доступу: URL: http://www.osp.ru/win2000/2004/05/177049/ (дата звернення 29.05.18).

49.Snort установка и минимальная настройка [Електроний ресурс]. – Режим доступу: http://www.kanava.biz.nf/mod/pub/article.php?art=710 (дата звернення25.05.18).

50.Учи IT [Електроний ресурс] – Режим доступу: URL: http://www.uchi-it.ru/2/1/7.html (дата звернення 10.04.18).