МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

МЕТОДИЧНІ ВКАЗІВКИ

ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

«ІНФОРМАЦІЙНА СТІЙКІСТЬ КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ ТА МЕРЕЖ»

ДЛЯ СТУДЕНТІВ ДЕННОЇ ФОРМИ НАВЧАННЯ

ЗІ СПЕЦІАЛЬНОСТІ

123 – «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

ЧАСТИНА ІІ

КРЕМЕНЧУК 2018

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Інформаційна стійкість комп'ютерних технологій та мереж» для студентів денної форми навчання зі спеціальності 123 – «Комп'ютерна інженерія» освітнього ступеня «Магістр». Частина ІІ

Укладачі: д. т. н., проф. М. І. Гученко,
к. т. н. П. П. Костенко,
асист. Н. Л. Сохін
Рецензент к. т. н., доц. О. Г. Славко

Кафедра комп'ютерних та інформаційних систем

Затверджено методичною радою Кременчуцького національного університету імені Михайла Остроградського
Протокол № _____ від _____
Голова методичної ради _____ проф. В. В. Костін

ЗМІСТ

ВСТУП

Методичні вказівки укладено на підставі робочої програми навчального курсу «Інформаційна стійкість комп'ютерних технологій та мереж» для студентів денної форми навчання зі спеціальності 123 – «Комп'ютерна інженерія» освітнього ступеня «Магістр».

Виконання лабораторних робіт дозволить студентам набути практичних навичок у галузі створення комп'ютерних технологій та мереж, стійких до дії негативних антропогенних та фізичних впливів, і використовувати сучасні інструментальні програмні засоби для їх проектування.

У результаті виконання лабораторних робіт студент повинен уміти:

– організовувати процес проектування та розроблення відмовостійкого програмного забезпечення;

– застосовувати методи забезпечення якості програмного забезпечення, її оцінювання та контролю;

– застосовувати методи забезпечення якості обслуговування користувачів (QoS);

– застосовувати методи керування потоками даних у комп'ютерних мережах;

– аналізувати загрози та застосовувати стандарти, що стосуються інформаційної безпеки комп'ютерних систем та мереж.

1 ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота № 5

Тема. Безпека Web-ресурсів

Мета: отримання практичних навичок використання методів та засобів збирання інформації про аналізований Web-додаток.

Короткі теоретичні відомості

Одним з перших етапів аналізу захищеності будь-якої комп'ютерної системи є збирання інформації. Залежно від використовуваної методології аналізу захищеності Web-програми можуть застосовуватися різні методи і засоби збирання інформації. Варто зазначити, що збирання інформації зазвичай не характерне для методології інструментального аналізу захищеності (сканування), а характерне для методології тестування на можливість проникнення.

Методи збирання інформації поділяють на активні і пасивні. Активні методи вимагають безпосередньої взаємодії з досліджуваним додатком за допомогою відправлення йому запитів і аналізу відповідних відповідей, а пасивні методи використовують інформацію, що відправляється сервером Web-додатка його клієнтам (наприклад, HTTP-заголовки X-Frame-Options, Strict-Transport-Security і т. д.) без відправлення запитів. Під час аналізу Web-додатків зазвичай використовують тільки активні методи.

Активні методи поділяють на методи з підключенням до додатка (наприклад, ідентифікація Web-сервера за допомогою сканера Httprint) і методи без підключення (наприклад, збирання інформації про програму пошуковими роботами, сканерами Інтернет і т. д.).

В результаті проведення збирання інформації про Web-додаток можуть бути отримані:

– імена і IP-адреси мережевих вузлів, на яких розміщені Web-додаток і його компоненти;

– логіни і паролі технологічних облікових записів;

– коментарі розробників;

– дані про системне і прикладне програмне забезпечення, що використовуються у засобах захисту та конфігурації Web-додатка;

– адреси електронної пошти розробників додатка;

– початковий код серверної частини Web-додатка;

– конфіденційні файли.

Програмними засобами отримання необхідної інформації є:

– пошукові системи (наприклад, Google, Shodan, Bing);

– спеціалізовані сканери вразливостей Інтернет (наприклад, http://un1c0rn.net/);

– інструментальні засоби аналізу захищеності мереж загального призначення (Nmap, Xprobe2, XSpider);

– інструментальні засоби аналізу захищеності мереж Web-додатків (AppScan, Acunetix, Burp Suite, ZAP, W3AF і т. д.).

Захист транспортного рівня Web-додатка грунтується на використанні протоколів сімейства SSL/TLS, що мають значну кількість механізмів, функцій та параметрів захисту, реалізація та конфігурація яких визначає рівень захищеності Web-додатка. Незважаючи на те, що на сьогодні відомо багато автоматизованих засобів тестування захищеності SSL/TLS (наприклад, сервіс www.ssllabs.com, програми SSLscan і SSLyze), деталі їх реалізації зазвичай невідомі або вимагають додаткового дослідження, що іноді не дозволяє повністю довіряти результатам їх роботи. Одним з низькорівневих і надійних засобів тестування захищеності служб SSL/TLS є клієнт OpenSSL.

Порядок виконання роботи

1. В адресному рядку браузера перейти за адресою *www.test.app.com/robots.txt*.

Проаналізувати вміст файлу. Зробити висновки про наявність «прихованих» директорій.

2. У адресному рядку браузера перейти за адресою *http://www.test.app.com/crossdomain.xml*, потім перейти за адресою *http://www.test.app.com/clientaccesspolicy.xml*. Проаналізувати вміст файлів. Зробити висновки про коректність конфігурації політики міждоменної взаємодії RIA.

3. Перейти за адресою *http://www.google.com*. Задати пошукові запити, які визначаються аналізованим додатком, наприклад:

– *site: www.test.app.com filetype: docx confidential*
– *site: www.test.app.com filetype: doc secret*
– *site: www.test.app.com inurl: admin*
– *site: www.test.app.com filetype: sql*
– *site: www.test.app.com intext: «Access denied»*

Проаналізувати логіку запитів і отримані дані. Побудувати свої запити, використовуючи приклади з бази запитів [14].

4. Перейти за адресою http://www.shodanhq.com. Задати такий пошуковий запит:

hostname: www.test.app.com

Побудувати свої запити для додатка www.test.app.com.

5. Даний тест виконується тільки для додатків, розміщених в лабораторній мережі. За допомогою мережевих сканерів Nmap і Xprobe виконати ідентифікацію ОС Web-сервера:

 # Nmap -O www.test.app.com -vv # xprobe2 www.test.app.com

6. Підключитися до Web-сервера, використовуючи утиліту Netcat:

 # Nc www.test.app.com 80

Надіслати такий GET запит:

GET / HTTP / 1.1

Host: www.test.app.com

\ r \ n

За заголовками Server і X-Powered-By визначити програмне забезпечення, що реалізує Web-сервер і фреймворк Web-програми.

У браузері встановити розширення Wappalyzer, перейти за адресою Web-додатка та проаналізувати інформацію про компоненти Web-додатка, отриману через Wappalyzer.

7. За допомогою сканера Web-серверів Httprint (дистрибутив Backtrack) або Httprecon (ОС Windows) виконати ідентифікацію Web-сервера:

 # Cd / pentest / enumeration / Web / httprint / linux

 # ./httprint -h www.test.app.com -s signatures.txt

За допомогою сканера Wafw00f перевірити наявність у Web-програми підсистеми WAF:

 # Cd / pentest / Web / waffit

 # Python ./wafw00f.py http: // www.test.app.com # python ./wafw00f.py https: // www.test.app.com

8. Виконати тести з ідентифікації підтримуваних Web-сервером HTTP-методів. Для цього необхідно відправити за допомогою Burp Suite або Netcat запит такого виду:

OPTIONS / HTTP / 1.1

Host: www.test.app.com

\ r \ n

Перевірити, чи підтримує сервер обробку запитів з довільними методами:

DOGS / HTTP / 1.1

Host: www.test.app.com

\ r \ n

Якщо Web-сервер підтримує метод TRACE, то це може призвести до уразливості до атаки Cross-Site Tracing (XST). Для перевірки підтримки Web-сервером методу TRACE відправити запит:

TRACE / HTTP / 1.1

Host: www.test.app.com

\ r \ n

Web-сервер підтримує метод TRACE і потенційно вразливий до атаки XST, якщо у відповідь на наведений вище запит буде отримано відповідь виду:

HTTP / 1.1 200 OK

Connection: close Content-Length: 39 TRACE / HTTP / 1.1

Host: www.test.app.com

9. Виконати базові перевірки SSL/TLS. Встановити останню версію пакета OpenSSL. Запустити мережевий аналізатор Wireshark. Виконати тестове підключення до сервера:

 # Openssl s_client -connect www.test.app.com:443

Переглянути трасування установки з'єднання в Wireshark. Визначити такі параметри: версію протоколу SSL/TLS, який використовується криптографічний набір (cipher suite), довжину відкритого ключа сервера, включення механізму стиснення даних. Відправити наступний HTTP-запит і переконатися в отриманні відповіді від сервера:

GET / HTTP / 1.1

Host: www.test.app.com

Перевірити підтримку сервером механізму «Sever Name Indication» (SNI):

 # Openssl s_client -connect www.test.app.com:443

-servername www.test.app.com

Переглянути трасування установки з'єднання в Wireshark в цьому випадку. Підтримка розширення SNI ідентифікується за допомогою установки з'єднання з опцією SNI і без неї. Якщо у відповідь отримані різні сертифікати, то SNI підтримується сервером. Якщо вказане в опції SNI ім'я невідоме, то клієнт виводить повідомлення про помилку або попередження.

10. Ідентифікувати всі підтримувані протоколи SSL/TLS, виконавши послідовно команди:

 # Openssl s_client -connect www.test.app.com:443

-ssl2

 # Openssl s_client -connect www.test.app.com:443

-ssl3

 # Openssl s_client -connect www.test.app.com:443

-tls1

 # Openssl s_client -connect www.test.app.com:443

-tls1_1

 # Openssl s_client -connect www.test.app.com:443

-tls1_2

Переглянути трасування сканування. Знайти відмінності в структурі мережевих повідомлень для різних версій протоколу.

11. Ідентифікувати криптографічні набори (cipher suite), підтримувані сервером. Для отримання всіх підтримуваних клієнтом криптографічних наборів виконати команду:

 # Openssl ciphers -v

Для перевірки підтримки, наприклад, набору AES256-SHA виконати таку команду:

 # Openssl s_client -connect www.test.app.com:443

-cipher AES256-SHA

Перевірити підтримку криптографічного набору, що містить шифр RC4:

 # Openssl s_client -connect www.test.app.com:443

-cipher RC4-SHA

Перевірити, що під час установки захищеного з'єднання криптографічний набір вибирається в порядку, визначеному налаштуваннями сервера, а не клієнта. Для цього зі списку підтримуваних сервером криптографічних наборів вибрати три вільних і виконати команди, наприклад:

 # Openssl s_client -connect www.test.app.com:443

-cipher 'AES256-SHA256, AES128-SHA, DES-CBC-SHA'

 # Openssl s_client -connect www.test.app.com:443

-cipher 'AES128-SHA256, AES256-SHA, DES-CBC-SHA'

 # Openssl s_client -connect www.test.app.com:443

-cipher 'DES-CBC-SHA, AES128-SHA, AES256-SHA256'

За правильного налаштування у всіх випадках має бути обраний набір AES256-SHA256.

Перевірити підтримку сервером Forward Secrecy на основі DHE і ECDHE:

 # Openssl s_client -connect www.test.app.com:443

-cipher 'ECDHE-RSA-AES256-SHA384'

 # Openssl s_client -connect www.test.app.com:443

-cipher 'DHE-RSA-AES256-SHA256'

12. Для визначення підтримки сервером механізму «Session Resumption» виконати команду:

 # Openssl s_client -connect www.test.app.com:443

-reconnect

або її менш інформативний варіант:

 # Openssl s_client -connect www.test.app.com:443

-reconnect | grep 'New \ | Reuse'

13. Для ідентифікації механізму «Secure Renegotiation» виконати команду:

 # Openssl s_client -connect www.test.app.com:443 | grep 'Secure Renegotiation'

Переглянути трасування сканування і переконатися в підтримці цього механізму. Підтримка механізму «Secure Renegotiation» сервером визначається за наявністю розширення «renegotiation_info» в повідомленні ServerHello або шляхом перегляду результату команди:

 # Openssl s_client -connect www.test.app.com:443

-tlsextdebug

Для ідентифікації підтримки сервером механізму «Client-Initiated Renegotiation» необхідно підключитися до Web-сервера по SSL/TLS за допомогою клієнта OpenSSL:

 # Openssl s_client -connect www.test.app.com:443

і потім відправити запит:

HEAD / HTTP / 1.1

Host: www.test.app.com R

або:

GET / HTTP / 1.1

Host: www.test.app.com R

Якщо сервер не підтримує «Client-Initiated Renegotiation», то буде виведено повідомлення про помилку. Якщо сервер підтримує цей механізм, то сервер знову відправить клієнту свої сертифікати.

Підтримка механізму «Client-Initiated Renegotiation» може бути використана для реалізації щодо Web-сервера DoS-атаки, так як за кожного встановлення з'єднання серверу потрібно витрачати значно більше обчислювальних ресурсів, ніж клієнту.

Щоб перевірити можливість реалізації DoS-атаки, можна перевірити, скільки раз клієнт може ініціювати переузгодження (renegotiation) криптографічних параметрів:

GET / HTTP / 1.1

Host: www.test.app.com

R

R

R

R

R

Інший спосіб тестування – використання експлойтів thc-ssl-dos [16], наприклад:

 # Thc-ssl-dos --accept 192.168.1.1 443

14. Перевірити наявність уразливості до атаки «BEAST». Ця атака використовує недоліки блокових шифрів, що працюють в режимі CBC, і існує в усіх версіях протоколів SSL/TLS до версії TLS 1.1. Для того, щоб захиститися від атаки BEAST, необхідно використовувати шифр RC4 або протокол TLS версії 1.1 і старше. Однак шифр RC4 на сьогодні вважається небезпечним, тому його використання є небажаним. Для захисту від атаки BEAST на практиці пропонується два способи: перший з них має назву «Суворе ослаблення» (Strict mitigation) і передбачає використання протоколу TLS версії 1.1 і старше з усіма клієнтами, які його підтримують; другий спосіб називається «Пріоритезація RC4» (RC4 prioritization) і полягає у підвищенні пріоритету шифру RC4 для

клієнтів, що підтримують тільки протоколи SSL 2.0, SSL 3.0 і TLS 1.0. Отже, необхідно переконатися, що клієнти SSL 3.0 або TLS 1.0, які не підтримують шифр RC4, не зможуть встановити з'єднання:

 # Openssl s_client -connect www.test.app.com:443

-no_ssl2 -no_tls1_1 -no_tls1_2 -cipher 'ALL:! RC4'

або що клієнти, що підтримують шифр RC4, встановлять з'єднання, використовуючи його:

 # Openssl s_client -connect www.test.app.com:443

-no_ssl2 -no_tls1_1 -no_tls1_2 -cipher 'ALL: + RC4'

15. Перевірити наявність уразливості до атаки «Heartbleed» за непрямими ознаками, а також за допомогою використання активних тестів в Metasploit Framework.

Переглянути трасування в Wireshark і визначити підтримку розширення «Heartbeat» після виконання команди:

 # Openssl s_client -connect www.test.app.com:443

-tlsextdebug

Перевірити підтримку сервером протоколу «Heartbeat» через OpenSSL за допомогою виконання команди:

 # Openssl s_client -connect www.test.app.com:443

-tlsextdebug

Якщо сервер не повертає в повідомленнях дані про розширення «Heartbeat», то він не вразливий до цієї атаки.

Для того щоб перевірити, чи відповідає сервер на запити Heartbeat, слід виконати команди:

 # Openssl s_client -connect www.test.app.com:443 -msg

Для перевірки вразливості клієнта (наприклад, Web-браузера) до Heartbleed-атаки можна встановити з'єднання з будь-яким сервером і переглянути трасування з'єднання.

Розглянемо варіант активного тестування (виконання атаки з використанням уразливості) в середовищі Metasploit Framework.

Для тестування клієнта виконати такі команди:

 # msfconsole

> *Use auxiliary / server / openssl_heartbeat_client_memory*

> *Show options*

> *run*

У Web-браузері відкрити ресурс Metasploit, що відповідає за тестування на наявність уразливості до атаки Heartbleed, і переглянути інформацію про результат тестування клієнта.

Для тестування сервера в середовищі Metasploit Framework виконати такі команди:

 # msfconsole

> *Use auxiliary / scanner / ssl / openssl_heartbleed*

> *Show options*

> *Set RHOSTS www.test.app.com*

> *Set RPORT 443*

> *Set VERBOSE true*

> *run*

За допомогою проекту *http://un1c0rn.net* знайти Web-сервери, вразливі до атаки Heartbleed (також можна використовувати тестові сервери, вразливі до атаки Heartbleed, наприклад, *heartbleed.csr-group.com*). Підтвердити вразливість до атаки в середовищі Metasploit Framework або за допомогою сервісу *http://ssllabs.com*.

16. Перевірити наявність уразливості до атаки «CRIME» за непрямими ознаками. Ця атака заснована на стисненні даних на рівні SSL/TLS. Для перевірки достатньо виконати команду:

 # Openssl s_client -connect www.test.app.com:443

-reconnect | grep 'Compression'

17. Перевірити наявність HTTP-заголовків Strict-Transport-Security, що встановлюються на стороні Web-сервера. Надіслати такий HTTP-запит до Web-додатка в програмі Burp Suite:

GET / HTTP / 1.1

Host: www.test.app.com

\ r \ n

HTTP-відповідь має містити заголовок такого виду:

Strict-Transport-Security: max-age = 31536000; includeSubDomains

Перевірити наявність сторінок зі змішаним контентом (mixed-content pages) – сторінок, доступних за HTTPS, але які містять ресурси (картинки, скрипти JavaScript, файли CSS, медіа-контент), доступні за протоколом HTTP. Для цього слід сконфігурувати браузер для роботи з тестованим Web-додатком через Web-проксі Burp Suite. В процесі роботи з Web-додатком необхідно у вкладці HTTP History переглянути історію і упевнитися, що всі запити до сервера відправляються лише за протоколом HTTPS.

Перевірити, що cookie, які містять чутливу інформацію, мають атрибут secure, наприклад:

Set-Cookie: SessionId = 371d2sm6cbn3d31a; path = /; secure

Перевірити, що чутливий контент не кешується на боці клієнта. Заборона кешування визначається наявністю HTTP-заголовків Pragma, Cache-Control і Expires з такими рекомендованими значеннями:

Pragma: no-cache

Cache-Control: no-cache, no-store, must-revalidate, max-age = 0

Expires: 0

Перевірити, що додаток захищено від атаки «SSL Stripping». Для цього необхідно переконатися, що Web-додаток доступний тільки за протоколом HTTPS і не є опціонально доступним за протоколом HTTP або в Web-додатку використовується заголовок Strict-Transport-Security (за умови того, що користувач гарантовано потрапить на сайт за протоколом HTTPS).

18. Виконати тестування SSL/TLS з використанням сервісу *ssllabs.com*.

У Web-браузері перейти за адресою *https://www.ssllabs.com/ssltest/index.html*, ввести доменне ім’я сервера, виконати сканування, переглянути і проаналізувати отримані результати.

Порівняти результати сканування сервера з результатами, отриманими під час його ручного тестування.

Виконати тестування декількох Web-клієнтів (наприклад, Web-браузерів Internet Explorer, Mozilla Firefox, Google Chrome, WhiteHat Security Aviator, Яндекс Браузер, Apple Safari, Opera тощо). Для цього в кожному тестованому браузері відкрити сторінку *https://www.ssllabs.com/ssltest/index.html*, виконати сканування, переглянути і проаналізувати результати.

19. Просканувати Web-сервери 10 відомих компаній, банків, операторів зв'язку за допомогою сервісу *ssllabs.com*. Проаналізувати результати.

Зміст звіту

1. Назва та мета роботи.

2. Опис виконання роботи.

3. Письмові відповіді на контрольні запитання.

Контрольні питання

1. Назвіть етапи аналізу захищеності Web-додатків.

2. Які дані можна отримати в результаті збирання інформації про Web-додаток?

3. Назвіть програмні засоби для збирання інформації про Web-додатки.

4. Атака Cross-Site Tracing (XST).

5. Вразливості механізму «Client-Initiated Renegotiation».

6. Атака «BEAST».

7. Атака «Heartbleed».

8. Атака «CRIME».

9. Атака «SSL Stripping».

Література: [5, 7–14].

Лабораторна робота № 6

Тема. Методи виявлення плагіату

Мета: ознайомитися з поняттям плагіату, його основними типами, розглянути найпоширеніші програми, призначені для виявлення плагіату.

Короткі теоретичні відомості

Плагіат **–** привласнення авторства на чужий твір науки, літератури, мистецтва або на чуже відкриття, винахід чи раціоналізаторську пропозицію, а також використання у своїх працях чужого твору без посилання на автора.

Єдиного, вичерпного та загальноприйнятого визначення плагіату не існує. Наприклад, автори одного з найбільших англомовних ресурсів для виявлення плагіату «Turnitin» (http://www.turnitin.com) наводять такі визначення цього поняття:

– видати за власні ідею або слова іншої людини ;

– використати результати роботи іншої людини без зазначення джерела, звідки вони були взяті;

– повністю або частково використати мистецький, науковий або інший твір чи роботу та видати їх за свою;

– подати вже існуючу ідею або продукт як новий та оригінальний.

Окрім того, в різних сферах і галузях діяльності плагіат може мати більш персоніфіковані відносно специфіки цієї діяльності визначення. Так, в навчальних закладах він може розглядатися як:

– офіційно не підтверджена домовленість між щонайменше двома студентами, наслідком якої є виконання однакових, або майже однакових, робіт;

– фальсифікація (зміст завдань, наприклад, статистичні показники, були вигадані або невірно вказані як результат власної роботи);

– реплікація (студент здає однакову, або майже однакову роботу кілька разів для підняття власного академічного рейтингу);

– використання не дозволених допоміжних матеріалів під час іспитів;

– отримання недозволеної копії екзаменаційних завдань;

– спілкування з іншими студентами під час іспитів для розв’язання завдань;

– видавання себе за іншого студента на іспитах.

Отже, плагіат в будь-якому разі розглядається як шахрайство, суть якого – у крадіжці чужої роботи або її частини і представленні її як власної. Загалом, можна поділити на три основні типи плагіату:

– копіювання чужої роботи (як без, так і з відома) та оприлюднення її під своїм іменем;

– подання суміші власних та запозичених в інших аргументів без належного цитування джерел;

– перефразування чужої роботи без належно оформленого посилання на оригінального автора або видавця.

Подібну, але більш розгорнуту класифікацію пропонують автори вже згаданого ресурсу «Turnitin»:

– видання виконаної іншим автором роботи як власної без внесення до неї жодних змін;

– копіювання значної частини чужої роботи до власної без внесення жодних змін;

– копіювання інформації з кількох різних джерел без внесення в неї правок. «Маскування плагіату» самостійним написанням перехідних речень між скопійованими частинами;

– внесення незначних правок у скопійований матеріал (переформулювання речень, зміна порядку слів в них тощо);

– повне запозичення текстів з інших джерел, але цілковите їх перефразування;

– видання власної колись вже написаної роботи за нову (переважно стосується студентів).

Найочевиднішими є перші типи. Їх найлегше виявити. Натомість,

перефразування чужої роботи та порушення правил цитування, що також є плагіатом, виявити значно важче. Тому цим часто зловживають.

Наслідками плагіату є зубожіння інформаційного простору, порушення законів, матеріальні та моральні збитки авторів, видавництв.

Для боротьби з плагіатом існує спеціальне програмне забезпечення та організації, працівники яких спеціалізуються на виявленні плагіату в науковій чи художній літературі за допомогою багаторівневого всебічного аналізу тексту.

Найпоширеніші програми для виявлення плагіату

– IN-SPIRE™ Visual Document Analysis (для опрацювання текстових даних науково-технічної літератури);

– VantagePoint (для глибинного аналізу тексту);

– Histcite (для формалізованого аналізу результатів пошуку в БД);

– Web of Science (містить ключові слова, авторів і процитованих авторів, журнали, країни й організації, від яких автори публікують свої статті).

– «Антиплагіат» (AntiPlagiat.ru).

Більшість програмних продуктів такого типу для виявлення плагіату використовують такі процедури:

– створення частотних масивів слів;

– визначення тематичного напряму тексту;

– динамічне визначення частотного словника певного тематичного напряму корпусу та дослідного тексту (статті) (вилучення стоп-слів, нормалізація тексту);

– визначення масивів слів: всі слова статті; всі слова корпусу; слова, які є в обох масивах; унікальні слова статті; унікальні слова корпусу;

– підрахунок косинусної міри.

Правила цитування

Цитування має використовуватися у всіх випадках, коли в роботі використовуються дані, взяті зі сторонніх джерел, а не отримані або створені

безпосередньо автором. Порушення чи недотримання вказаних нижче правил має розцінюватися як плагіат:

– якщо думка автора наводиться дослівно, то її слід взяти в лапки;

– якщо цитується великий уривок тексту, то він може не братися в лапки, натомість – виділяється або відокремлюється від решти тексту певним способом (набирається іншим кеглем, шрифтом, накресленням, відокремлюється від основного тексту більшими абзацними відступами тощо);

– допускається скорочення цитати, яке не призводить до викривлення думки автора. Місце скорочення має бути позначене в цитаті квадратними дужками з трикрапкою всередині;

– допускається перефразування цитати, зміна словоформ чи відмінків певних слів. В такому разі, цитата в лапки не береться, але в квадратних дужках обов'язково ставиться посилання на джерело (його порядковий номер зі списку використаної літератури, який додається до роботи);

– в списку використаної літератури завжди слід вказувати навіть ті джерела, які використовувалися під час підготовки роботи і вивчення теми, навіть якщо прямих посилань чи цитувань цих джерел в роботі нема.

З розвитком програмування набув поширення ще один вид плагіату – видання програмного коду, написаного іншою людиною, за власний.

Розглянемо основні детектори такого плагіату: MOSS, JPlag, SIM, Sherlock (BOSS), PMD(CMD), CodeMatch.

MOSS – автоматична безкоштовна система пошуку плагіату, яка має дуже простий інтерфейс і працює он-лайн. Для використання програми, користувачеві необхідно лише вказати список файлів, які необхідно перевірити. У відповідь на запит сервер MOSS створює HTML сторінки з переліком пар програм зі схожими частинами коду, які виділені підкресленням. Результати сканування коду можуть бути доступними для перегляду протягом 14 днів. Користувач може ввімкнути функцію автоматичного видалення коду, що був автоматично згенерований, або код бібліотек, щоб бачити яку кількість коду програміст написав самостійно.

Детектор плагіату MOSS здатен аналізувати код, що написаний мовами: C, C++, Java, C#, Python, Visual Basic, JavaScript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula-2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS та HCL2. Така велика кількість підтримуваних мов програмування є значною перевагою порівняно з іншими детекторами плагіату. Для того, щоб розпочати користування цією системою, необхідно відправити свої контактні дані на офіційну електронну пошту сервісу.

Але, незважаючи на всі переваги, система є закритою, що не дозволяє переглядати алгоритми, які використовуються у програмі, а відповідно не дозволяє дізнатись, які техніки плагіату може знайти ця система, а також забороняє додати можливість пошуку коду іншими мовами, які на сьогодні MOSS не підтримує.

JPlag – найбільш популярний безкоштовний детектор плагіату, який дозволяє порівнювати код програми з наявною базою вже написаних програм.

JPlag підтримує невелику кількість існуючих мов програмування: Java, C#, C, C++, Scheme, але має декілька переваг. Серед них можливість пошуку плагіату серед текстів, написаних природною мовою, та використання для нових та універсальних алгоритмів пошуку плагіату.

Для початку користування програмою необхідно зареєструватися на офіційному сайті. Результати сканування коду програм доступні доти, доки користувач не видалить історію пошуків. Інтерфейс програми є зрозумілим і дозволяє не тільки знайти і підкреслити частини коду, в яких помічено плагіат, але й надає гістограми, на яких показано кількість та частота випадків копіювання. Система не є відкритою, але незважаючи на це розробники надали інформацію про використовувані алгоритми, методики пошуку та види технік плагіату, які може виявити система JPlag.

SIM – детектор плагіату, який перевіряє код за допомогою вимірювання лексичної схожості текстів, написаних мовами C, Java, Pascal, Modula-2, Miranda. Також ця система дозволяє перевіряти тексти, написані природною мовою. Детектор плагіату SIM працює локально, не вимагаючи підключення до

Інтернет. Головною перевагою цієї системи є те, що вона є однією з небагатьох безкоштовних та відкритих детекторів коду, а також знаходиться в постійній розробці та вдосконалюється.

Одним з недоліків цієї системи є незручний інтерфейс та відсутність графічного та достатнього статистичного подання результатів з виконаного пошуку.

Sherlock (BOSS) **–** частина системи BOSS, яка може працювати як окремий додаток. Цей детектор плагіату порівнює вихідний код і тексти, написані природною мовою. Система Sherlock є відкритою та безкоштовною, але, незважаючи на це, підтримка користувачів розробниками відсутня та можливості модифікації програми власноруч для додавання нових мов немає.

Головним недоліком цієї системи є підтримка невеликої кількості мов програмування: Java, C та C++.

Детектор плагіату аналізує код програм на найпопулярніші техніки плагіату: додавання (видалення) пробілів, додавання (видалення) коментарів, перестановка частин коду у тексті та перейменування змінних. Але, не зважаючи на це, Sherlock використовує застарілі версії алгоритмів, що не дозволяє отримати якісні результати пошуку плагіату.

Інтерфейс програми є достатньо зрозумілим, а статистичної інформації вистачає для того, щоб виявити плагіат (графіки, таблиці зі статистикою та можливість явно передивитися виділені кольором частини в коді, які були скопійовані).

PMD (CMD) – відкрита та безкоштовна система виявлення плагіату, яка не була задумана, як інструмент пошуку плагіату коду в сфері навчання, але зараз добре працює і в цій сфері. Дозволяє знаходити плагіат в коді програм, написаних мовами Java, JSP, C, C++, PHP і FORTAN. Головною з переваг цієї системи є існування можливості та інструкцій для додавання нової мови програмування, для аналізу коду, написаного нею.

Результати аналізу коду програма повертає у файл з аналізованим кодом програми. Це незручно, адже наочно не можна побачити, які частини коду

скопійовані, недостатньо статистики і робота з програмою проводиться з командного рядка.

PMD сканує код на виявлення таких основних технології плагіату:

- змінні, що не використовуються;
- перейменовані змінні;
- копіювання приватних методів;
- додавання (видалення) коментарів;
- дублювання коду;
- завантаження з Інтернет.

PMD інтегрований з такими середовищами розробки, як JDeveloper, Eclipse, Jedit, JBuilder, BlueJ, CodeGuide, IntelliJ IDEA, TextPad, Maven, Ant, JCreator, і Emacs.

CodeMatch – це комерційна програма-детектор плагіату, що має дуже широкі можливості для аналізу коду програм. CodeMatch на сьогодні підтримує такі мови програмування: Basic, C, C++, C#, Delphi, Flash ActionScript, Java, JavaScript, MASM, Pascal, Perl, PHP, PowerBuilder, Ruby, SQL, Verilog, VHDL [4].

В пакеті з CodeMatch можуть бути додаткові програми перевірки на плагіат (BitMatch та CodeDiff). Детектор CodeMatch порівнює тільки вихідний код, на відміну від BitMatch, який корисний у разі, якщо є тільки доступ до виконуваних файлів, а не до вихідного коду. CodeDiff корисний, якщо вже точно відомо, що існує плагіат у файлах, які будуть перевірятися. CodeDiff надає більш детальну інформацію про те, що і як було скопійовано та перероблено.

CodeMatch порівнює кожен файл із вказаних каталогів, включаючи всі підкаталоги, якщо потрібно. CodeMatch створює бази даних з перевірених програм. Інформація, про виконання пошуку відображається на HTML-сторінках, як звіт. Можна обрати, наскільки докладний звіт буде наданий.

В CodeMatch на сьогодні підтримуються такі мови програмування: ABAP,

ASM-m68k, BASIC, C++, C#, Delphi, Flash, ActionScript, Fortran, FoxPro, Java, JavaScript, LISP, MASM, MATLAB, Pascal, Perl, PHP, PL/M PowerBuilder, Python, REALbasic, SQL, Verilog, VHDL, Visual Basic тощо. Підтримка нових мов є простою та швидкою. Якщо користувачеві необхідно проаналізувати код мовою, якої в переліку немає, то розробники додадуть дану мову протягом декількох днів.

Порядок виконання роботи

1. Запустіть Інтернет-браузер.
2. Перейдіть на сайт http://www.antiplagiat.ru.
3. Користуючись інформацією, що знаходиться на цьому сайті, запишіть до звіту:
   - набір послуг, що надає даний сервіс, їх ціна, наявність тарифів;
   - опис технологій, які використовуються для виявлення плагіату.
4. Перевірте будь-який фрагмент тексту на оригінальність. Результати запишіть до звіту.
5. Зробіть та запишіть до звіту висновки, в яких зробіть аналіз цього сервісу, зазначте його переваги та недоліки.

Зміст звіту

1. Назва та мета роботи.
2. Опис виконання роботи.
3. Письмові відповіді на контрольні запитання.

Контрольні питання

1. Що охоплює поняття плагіату? Назвіть різні варіанти тлумачення цього терміну.
2. Назвіть та охарактеризуйте найпоширеніші програми для виявлення плагіату.
3. Зробіть аналіз програми «Антиплагіат».
4. Які процедури використовують у своїй роботі програми для виявлення плагіату?
5. Назвіть основні правила цитування.

6. Охарактеризуйте програми для виявлення плагіату в програмному коді.

7. Яку шкоду несе в собі явище плагіату?

8. Як за законодавством України карається плагіат?

Література: [4, 6].

Лабораторна робота № 7

Тема. Попередження вторгнень у комп'ютерні мережі

Мета: отримання практичних навичок роботи із засобами виявлення мережевих атак і сканерів портів (сніфферів) у комп'ютерній мережі.

Короткі теоретичні відомості

Основним призначенням утиліти *APS* є виявлення мережевих атак. Утиліта *APS* проводить обмін з атакуючим і дозволяє однозначно ідентифікувати факт атаки, тому що після сканування портів багато сканерів роблять визначення типу сервісу за допомогою передачі тестових запитів і аналізу відповіді сервера. Утиліта *APS* призначена:

– для виявлення атак сканування портів, ідентифікації сервісів, появи в мережі троянських програм, мережевих «хробаків». У базі *APS* більше сотні портів, використовуваних «хробаками» та Backdoor-компонентами;

– для тестування сканерів портів і мережевої безпеки. Для перевірки роботи сканера необхідно запустити на тестовому комп'ютері *APS* і провести сканування портів. За протоколами *APS* можна встановити, які перевірки проводить сканер і в якій послідовності;

– для тестування і контролю за роботою *Firewall* утиліта *APS* запускається на комп'ютері із установленим *Firewall* і проводиться сканування портів або інших атак. Якщо *APS* видає сигнал тривоги, то це є сигналом про непрацездатність *Firewall* або про його невірне налаштування. *APS* може бути постійно запущена на захищеному за допомогою *Firewall* комп'ютері для контролю за справним функціонуванням *Firewall* у реальному часі;

– блокування роботи мережевих «хробаків» і *Backdoor* модулів. Принцип виявлення й блокування заснований на тому, що той самий порт може бути відкритий на прослуховування тільки один раз, тому відкриття портів, використовуваних троянськими й Backdoor програмами до їхнього запуску перешкодить їхній роботі, а після запуску призведе до виявлення факту використання порту іншою програмою;

– тестування антитроянських і антивірусних програм, систем *IDS*. У базі *APS* закладено більше сотні портів найпоширеніших троянських програм. Деякі антитроянські засоби мають здатність проводити сканування портів вузла, що перевіряється, або будувати список портів, що прослуховуються, без сканування за *допомогою* API Windows. Такі засоби мають повідомляти про підозру на наявність троянської програми з виводом списку портів.

У переданому банері можливі макроси, які будуть замінені в момент відправлення на їхні значення. На сьогодні підтримуються такі макроси:

– *#DATE#* – заміняється на поточну дату, відформатовану відповідно до налаштувань системи, формат DD.MM.YYYY;

– *#TIME#* – заміняється на поточний час, відформатований відповідно до налаштувань системи, формат HH24.MI.SS;

– *#DATETIME#* – заміняється на поточну дату й час, відформатовані відповідно до налаштувань системи, формат DD.MM.YYYY HH24.MI.SS;

– *#UNIXDATE#* – дата у форматі, прийнятому в Інтернет;

– *#UNIXDATETIME#* – дата і час у форматі, прийнятому в Інтернет;

– *#RAND_BIN#* – випадкові бінарні дані випадкової довжини, мінімальна довжина блоку даних становить 50 байт, максимальна – 250;

– *#RAND_TXT#* – випадкові текстові дані випадкової довжини, мінімальна довжина блоку даних становить 50 байт, максимальна – 250. Відрізняється від *#RAND_BIN#* тим, що складається з байтів з кодами 32–127, текст, цифри й пропуски;

– *$xx* – байт, *xx* – значення в шістнадцятирічному форматі, для символів з кодами 0–9 обов'язковий попередній нуль, тобто *$00, $01*. Застосовується для

введення у переданий текст бінарних даних, у першу чергу для передачі символів переходу на новий рядок та переведення каретки ($0D і $0A).

Параметри *#TIME#, #DATETIME#, #UNIXDATE#, #UNIXDATETIME#* застосовуються для додання відповідям *APS* реалістичності. Сучасні сканери мають інтелект і розуміють, що замість сервісу встановлена *APS* – це досягається порівнянням банерів, отриманих у результаті декількох підключень до порту. Параметри *#RAND_BIN# і #RAND_TXT#* ускладнюють роботу інтелектуального сканера, тому що він намагається аналізувати отримані дані.

Порядок виконання роботи

1. Запустіть сканер портів *XSpider* на сканування сусідніх робочих станцій з виявленням можливості *DoS*-атак.

2. Запустіть утиліту *APS* для виявлення факту сканування портів за протоколами *TCP, UDP* і розсилання *UDP broadcast* пакетів для заданих портів.

3. Налаштуйте утиліту *APS*. Для цього з пункту меню «*Сервіс/Налаштування*» викличте діалогове вікно «*Налаштування*».

4. Відкрийте вкладку «*Загальні налаштування*», категорію «*Інтерфейс*». Ознайомтесь з можливостями цього меню. Відключіть перемикач звукового сигналу у разі виявлення атаки й задайте опцію розкриття вікна у разі виявленя атаки. Включіть перемикач «*Сортування списку портів після їхнього завантаження*» для автоматичного сортування бази портів за номером порту. Задайте автоматичне стартування програми у разі старту системи.

5. Відкрийте вкладки «*Оповіщення*», «*Звіти*», «*Протоколювання*» призначені, відповідно, для налаштування оповіщення адміністраторів за мережею, що проводиться за допомогою відправлення повідомлень на *mailslot* з ім’ям «*messngr*» і є аналогом виконання команди *NET SEND*, для налаштування передачі інформації службі *SysLog* серверів, для передачі звітів електронною поштою, для налаштування системи протоколювання утиліти. Ознайомтесь з можливостями цього меню. Зробіть налаштування відповідно до завдання.

6. Для зміни складу інформації повідомлень і листів, які відправляються утилітою *APS*, створіть шаблон, який формує звіт у форматі *XML*, з

необхідними даними; шаблон, що формує мінімальний за обсягом звіт для відправлення по *SMS*, з необхідними для вас даними; шаблон, що формує таблицю *CSV* формату для аналізу в *Excel*, з необхідними для вас даними. Шаблони зберігаються в текстовому файлі з розширенням *etf* у директорії *template*. Формат шаблону:

[Info]

Name=назва шаблону

[Header]

...

[Footer]

...

[HackerInfo]

...

[HackerPortInfo]

...

Додаткові відомості про параметри секцій шаблону

Секція *Info* містить обов'язковий параметр *Name=<назва шаблону>* і параметр *MaxRecCount*, що задає максимальну кількість записів, виведених у лист. Секції *Header* і *Footer* містять дані, що одноразово виводяться на початку і наприкінці листа. У цих секціях, крім довільного тексту, припустимі макроси, які замінюються в момент генерації листа значеннями. Ці макроси припустимі в будь-якій секції:

#VERSION# – версія програми;

#DB_VERSION# – дата відновлення й версія бази портів;

#TIME# – час формування листа;

#LOCAL_IP# – IP адреса комп'ютера, на якому спрацював *APS;*

#LOCAL_HOST# – ім'я комп'ютера, на якому спрацював *APS.*

Секція *HackerInfo* містить шаблон, за яким формуються дані кожного з атакуючих вузлів. У цій секції допустимі макроси, що замінюються в момент

генерації листа значеннями:

#HACKER_IP#, IP – адреса атакуючого комп'ютера;

#HACKER_HOST# – ім'я вузла для атакуючого *IP*;

#ATTACK_START_DATE# – дата початку атаки;

#ATTACK_START_TIME# – час початку атаки;

#ATTACK_END_DATE# – дата завершення атаки;

#ATTACK_END_TIME# – час завершення атаки;

#ATTACK_COUNT# – кількість атак;

#ATTACK_DOS_COUNT# – кількість підозр *DoS*;

#ATTACK_PORT_COUNT# – кількість атакованих портів;

#ATTACK_PORT_COUNT# – кількість атакованих портів;

#ATTACK_PORT_DETAIL# – деталізовані дані про порти;

#ATTACK_EXPRESS_TEST# – результати експрес-оцінювання, чи є *DoS, Flood*, сканування, виводиться в текстовому вигляді в три рядки.

Секція *HackerInfo* містить шаблон, за яким формуються дані кожного з атакованих портів. У цій секції допустимі макроси, що замінюються в момент генерації листа значеннями:

#HACKER_IP# – *IP*-адреса атакуючого комп'ютера;

#HACKER_HOST# – ім'я вузла для атакуючого *IP*;

#PORT_NUM# – номер порту;

#PORT_PROTO# – протокол *TCP* або *UDP*;

#PORT_ATTACK_COUNT# – кількість атак по даному порту;

#PORT_DOS_COUNT# – кількість підозр *DoS* по даному порту;

#PORT_DATA_SIZE# – обсяг даних, отриманих по даному порту від атакуючих; *#PORT_FLOOD_INFO#* – ознака флуда в текстовому вигляді.

7. Натисніть кнопку «*Тестувати налаштування*», для тестування правильного налаштування відправлення оповіщень за мережею. У разі натискання цієї кнопки програма відправить тестове повідомлення відповідно до поточних налаштувань.

8. Відкрийте вкладку «*Імітація сервісів*» для налаштування системи

імітації сервісів *TCP*. Ознайомтесь з можливостями цього меню. Зробіть налаштування відповідно до завдання. Зробіть налаштування імітації сервісів *UDP*, настроївши передачу випадкових даних замість відгуку з бази даних.

9. Відкрийте вкладку «*Редагування користувальницької бази портів*». Ознайомтесь з можливостями функцій, наведених у вікні редактора бази портів. У стовпці «*Переданий текст*» утримується текст банера. Задана в цьому полі інформація передається атакуючому вузлу після підключення до порту. Ознайомтесь з призначенням динамічних елементів, що утримуються, у даному стовпці.

10. Включіть перемикач «*Використовувати оповіщення по мережі NET SEND*», для активування режиму оповіщення за мережею. Заповніть поле «*ПК адміністраторів*», включивши імена *NetBios* комп'ютерів адміністраторів. Задайте дії утиліти *APS* під час з'єднання з атакуючим портом для передачі відгуку з бази даних *APS* атакуючому.

11. Дозвольте доступ до вбудованого Web-сервера формувача звітів з IP-адрес сусідніх комп'ютерів. Виключіть систему імітації для того, щоб не проводити активної взаємодії з атакуючим і негайно розривати з'єднання з ним.

12. Включіть перемикач «*Використовувати запис в Syslog на зазначених серверах*» для керування режимом відправлення інформації служби *Syslog*. Занесіть у поле «*Адреси серверів*» список IP-адрес серверів мережі. Задайте дії утиліти *APS* під час з'єднання з атакуючим портом, для передачі відгуку атакуючому з бази даних *APS*, плюс блок даних змінної довжини, від 100 до 500 байт, заповненого випадковими даними.

13. Налаштуйте режим відправлення звіту електронною поштою. Налаштуйте дії програми після обміну з атакуючим, утримуючи з'єднання відкритим за відсутності *Flood* і розриваючи за його наявності (поріг становить більше 60 підключень за хвилину).

14. Налаштуйте оповіщення «*тривоги*» електронною поштою. Налаштуйте передачу відгуку із заданою імовірністю за допомогою регулятора імовірності, щоб внести додатковий фактор випадковості й спотворити

результати сканування.

15. Задайте відправлення повідомлення для кожної події. Налаштуйте дії програми після обміну з атакуючим, розриваючи з'єднання, після обміну з атакуючим з ініціативи утиліти *APS*.

16. Налаштуйте пункт меню «*Оповіщення*» таким чином, щоб адміністратор одержував повідомлення від програми тільки на локальному комп'ютері. Включіть режим утримання з'єднання для утримання з'єднання з атакуючим, що сповільнить роботу деяких сканерів мережевої безпеки.

17. Включіть перемикач «*Розкривати вікно програми у разі виявлення атаки*» для автоматичного відображення головного вікна програми у разі виявленні спроби сканування портів. Налаштуйте дії *APS* під час з'єднання з атакуючим портом таким чином, щоб атакуючому передавався буфер, заповнений випадковими байтами.

18. Налаштуйте деталізацію листів, указавши відправлення списку атакуючих і даних про порти. Налаштуйте дії *APS* під час з'єднання з атакуючим портом для передачі текстового рядка, заповненого випадковими символами.

19. Задайте правила найменування й ротації протоколів. Налаштуйте дії програми після обміну з атакуючим, утримуючи з'єднання відкритим тривалий час.

20. Налаштуйте відправлення повідомлень для кожної події сканування портів, що містять не статистику про атакуючих, а дані про кожне конкретне сканування. За допомогою перемикача «*Включити емуляцію сервісів TCP*» виключіть систему імітації сервісів *TCP* для негайного розриву з'єднання з атакуючим вузлом без передачі атакуючому вузлу будь-якої інформації.

21. Задайте інтервал між повідомленнями з інформацією про атаку в числове поле «*Відправляти повідомлення не частіше, ніж один раз в XXX хвилин*». Налаштуйте режим передачі випадкових даних, тому що наявність у відповіді *APS* випадкової інформації вводить в оману сканери мережевої безпеки, ускладнює та сповільнює їх роботу.

22. Після налаштування утиліти *APS* запустіть її знову та переконайтеся, що налаштування утиліти працюють.

Зміст звіту

1. Назва та мета роботи.

2. Опис виконання роботи.

3. Письмові відповіді на контрольні запитання.

Контрольні питання

1. Для чого призначена утиліта APS?

2. Переваги утиліти APS.

3. Для чого призначений фільтр утиліти APS?

4. Що містить закладка налаштування системи імітації сервісів UDP? Для організації яких атак може застосовуватися відгук за UDP-портами при включенні імітації сервісів UDP?

5. Поясніть призначення користувальницької бази портів.

6. Які варіанти можливі при виборі протоколу у вікні редактора користувальницької бази портів?

Література: [1–3, 17].

Лабораторна робота № 8

Тема. Особливості криптографії у віртуальних системах

Мета: отримати практичні навички використання методів та засобів захисту даних у віртуальному середовищі.

Короткі теоретичні відомості

Поява віртуальних систем змінила форму загроз. З одного боку у віртуальному середовищі зникають загрози «реального» світу (наприклад, розрив кабелю, вихід з ладу плати конкретного сервера тощо), але виникають інші (наприклад, некоректне програмне налаштування логічних зв’язків між об’єктами віртуального середовища, можливість крадіжки/знищення віртуального середовища цілком завдяки тому, що віртуальні машини являють

собою файли, які можна скопіювати на знімний носій або видалити). Стандартні поняття ІТ (сервер, кабель, мережевий комутатор тощо) з фізичних об’єктів починають перетворюватися на віртуальні. У середовищі віртуалізації вони починають являти собою елементи налаштування програмного коду віртуального середовища (гіпервізора).

За архітектурними рівнями загрози безпеці у віртуальному середовищі поділяють на загрози:

– апаратній платформі, на якій розгортається віртуальне середовище;

– системному ПЗ віртуалізації (гіпервізору), яке виконує функції управління апаратними ресурсами і ресурсами віртуальних машин;

– системі управління віртуальним середовищем (серверні і клієнтські програмні компоненти, що дозволяють локально або дистанційно керувати настройками гіпервізора і віртуальних машин);

– віртуальним машинам, що включають в себе системне і прикладне програмне забезпечення;

– мережі зберігання даних (що включають комутаційне обладнання і систему зберігання) з розміщеними образами віртуальних машин і даними (мережа зберігання даних може бути реалізована на основі SAN, NAS, iSCSI).

Під час аналізу загроз важливо враховувати специфіку обробки персональних даних у віртуальному середовищі:

– дані обробляються в рамках віртуальних машин. В рамках одного сервера може існувати безліч віртуальних серверів, на кожному з яких можуть оброблятися персональні дані (ПД) різних категорій, а самі сервери можуть входити в різні інформаційні системи персональних даних. При цьому «образ» сервера може перебувати в окремому сховищі даних;

– дані передаються між віртуальними машинами всередині віртуального середовища. Оскільки віртуальна машина – це файл, що зберігається в сховищі даних, то передача персональних між віртуальними машинами передбачає, що персональні дані виходять з однієї області сховища даних, проходять мережеві

комутатори, потрапляють на сервер з розгорнутим віртуальним середовищем, а потім у зворотному порядку повертаються до сховища, але вже для іншої віртуальної машини;

– дані передаються між віртуальним середовищем і зовнішніми середовищами (як реальними, так і віртуальними). Особливість полягає в тому, що з мережевого інтерфейсу одного фізичного сервера можуть «виходити» дані, що відносяться до різних віртуальних серверів або інформаційних систем і, відповідно, виникає питання інтеграції механізмів захисту віртуального середовища і зовнішніх фізичних компонентів.

Розглянемо загрози кожного з рівнів, викликані навмисними або ненавмисними діями потенційних порушників.

Загрози апаратній платформі (звичайні серверні платформи, blade-кошики):

– порушення роботи апаратних компонентів серверного обладнання з встановленими компонентами віртуального середовища;

– несанкціоноване виймання з серверного обладнання і крадіжка носіїв інформації;

– порушення мережевої комутації.

Загрози системному ПЗ віртуалізації (Гіпервізор):

– некоректне налаштування параметрів гіпервізора і віртуальних машин, що впливають на безпеку (приклади можливих загроз і відповідних їм налаштувань безпеки для VMware vSphere 4.1 описані в «Security Hardening Guide» https://communities.vmware.com/docs/DOC-15413);

– помилки в роботі ПЗ гіпервізора;

– підміна виконуваних модулів ПЗ гіпервізора;

– несанкціонований віддалений доступ до ресурсів гіпервізора внаслідок мережевих атак типу «переповнення буфера» на відкриті мережеві порти сервера з гіпервізором в разі виникнення в його ПЗ вразливостей;

– виснаження обчислювальних ресурсів сервера з гіпервізором

внаслідок атак типу «відмова в обслуговуванні» щодо віртуальних машин;

– випадкове або навмисне перекручування/знищення образів віртуальних машин.

Загрози системі управління віртуальним середовищем:

– отримання несанкціонованого доступу до консолі управління віртуальною інфраструктурою (АРМ адміністратора віртуального середовища);

– отримання несанкціонованого доступу до налаштувань віртуальних машин;

– отримання несанкціонованого віддаленого доступу до інтерфейсу системи управління.

Загрози IT-інфраструктурі, реалізованій в межах віртуального середовища:

– розгортання нових погано захищених віртуальних машин;

– «змішування» інформації різного рівня конфіденційності в рамках єдиної апаратної платформи;

– несанкціоноване мережеве підключення до віртуальної машини;

– підміна та/або перехоплення даних і оперативної пам’яті віртуальних машин в процесі їх міграції засобами віртуального середовища;

– проведення мережевих атак між віртуальними машинами;

– вірусне зараження віртуальних машин.

Загрози мережі зберігання даних з розміщеними образами віртуальних машин:

– крадіжка розділів системи зберігання з образами віртуальних машин і даними;

– крадіжка носіїв даних мережі зберігання;

– фізичне знищення носіїв даних мережі зберігання;

– отримання несанкціонованого доступу до АРМ адміністратора мережі зберігання;

– отримання несанкціонованого доступу (віддаленого) до керуючих

інтерфейсів компонентів мережі зберігання;

– отримання несанкціонованого доступу з віртуальних машин, де обробка ПД не дозволена, до розділів системи зберігання з розміщеними ПД.

Одним з найбільш простих і одночасно найпотужніших методів контролю вхідного доступу є фільтрація TCP/IP. Фільтрація TCP/IP корисна з погляду безпеки, оскільки працює в режимі ядра, на відміну від протилежних методів контролю вхідного доступу на комп'ютери, наприклад, фільтри політики IPSec або сервер маршрутизації й віддаленого доступу, які залежать від процесів режиму користувача або служби робочих станцій і серверів.

Для контролю вхідного доступу за протоколом TCP/IP може бути використана комбінована схема із застосуванням фільтрації TCP/IP, фільтрів IPSec і фільтрації пакетів маршрутизації й віддаленого доступу. Такий метод особливо ефективний, якщо потрібно контролювати як вхідні, так і вихідні пакети протоколу TCP/IP. Фільтрація TCP/IP дозволяє стежити тільки за вхідним доступом.

Internet Protocol Security (IPSec) – це набір протоколів шифрування, аутентифікації й забезпечення захисту під час транспортуванні IP-пакетів. IP Security надає можливість шифрування всього мережевого трафіка на третьому рівні, що дає можливість використання небезпечних протоколів, таких, наприклад, як *telnet*, оскільки всі дані будуть інкапсульовані утилітою *IPSec* і зашифровані при передачі мережею.

Існує два режими роботи *IPSec*: транспортний і тунельний.

У транспортному режимі шифрується або підписується тільки інформативна частина IP-пакета. Маршрутизація не зачіпається, тому що заголовок IP-пакета не змінюється, не шифрується. Транспортний режим зазвичай використовується для встановлення з'єднання між вузлами. Він може також використовуватися між шлюзами, для захисту тунелів, організованих яким-небудь іншим способом, IP tunnel, GRE.

У тунельному режимі IP-пакет шифрується цілком. Для того, щоб його можна було передати за мережею, він поміщається в інший IP-пакет – це

захищений IP-тунель. Тунельний режим може використовуватися для підключення віддалених комп'ютерів до віртуальної приватної мережі або для організації безпечної передачі даних через відкриті канали зв'язку, наприклад, Інтернет, між шлюзами для об'єднання різних частин віртуальної приватної мережі.

Режими *IPSec* не є взаємовиключними. На тому самому вузлі можливе використання транспортного й тунельного режиму.

Шифрування даних для будь-якого додатка відбувається перед передачею їх мережею. Під час пересилання шифрованих даних клієнтським додатком з комп'ютера 1 на серверний додаток на комп'ютері 2 додатки на обох сторонах з'єднання ніяк не беруть участі у процесі шифрування. Клієнтський додаток посилає дані нижче по стеку протоколів, де вони перехоплюються протоколом *IPSec*, шифруються й потім посилаються на сервер. На сервері дані рухаються вгору по стеку протоколів і розшифровуються *IPSec* перед їхньою передачею серверному додатку.

IPSec може бути як службою шифрування, так і аутентифікації.

Функція забезпечення безпеки мережевого трафіка за допомогою шифрування здійснюється за допомогою протоколу, що має назву *ESP* (*Encapsulating Security Payload*).

Іншою функцією *IPSec* є здатність перевірки дійсності й цілісності пакетів за допомогою протоколу *AH* (*Authentication Header*). Цей протокол позначає пакети спеціальним підписом так, що одержувач може бути впевнений, що дані надійшли від справжнього відправника. Протокол забезпечує захист даних від змін під час передачі, так що ніхто не міг підмінити пакети. Протокол *AH* забезпечує цілісність даних, але не робить нічого для безпеки потоку даних, якщо не використовується разом з *ESP*.

Трафік *ESP* використовує IP-порт 50, протокол AH використовує IP-порт 51. Якщо два комп'ютери хочуть використовувати *IPSec* для цілей безпечної взаємодії, вони повинні бути настроєні на використання політики *IPSec*. Ці політики настроюються за допомогою локальних або групових політик ОС

Windows. Політики *IPSec* повинні визначати протоколи, які потрібно захищати, визначати використання ключів, визначати механізми аутентифікації.

За замовчуванням існує три політики IPSec. Розташовані в директорії \Windows Settings\Security Settings\IPSec Policies, ці політики є відповідно політиками для Client (Respond Only) (Клієнт – тільки відповідь), Secure Server (Require Security) (Безпечний сервер – вимагати безпеку) і Server (Request Security) (Сервер – запит безпеки). Тільки одна політика може бути застосована до системи в даний момент часу.

Порядок виконання роботи

1. Запустіть консоль керування IPSec на вашому комп'ютері. Виберіть одну з політик IPSec, використовувану за замовчуванням. Для цього відкрийте меню «Локальна політика безпеки» (Local Security Policy) у директорії «Адміністрування» (Administrative Tools) і виберіть розділ «Політики безпеки IP на локальному комп'ютері» (IP Security Policies on Local Computer). Ознайомтесь з принципами цієї політики безпеки.

2. Створіть свій список фільтрів. Для цього виберіть у вікні «Локальні налаштування безпеки» (Local Security Settings) і виберіть у контекстному меню «Дія» пункт «Керування списками IP-фільтра й діями фільтра» (Manage IP filter lists and filter actions). На закладці «Керування списками фільтрів IP» (Manage IP Filter Lists) натисніть кнопку «Додати» (Add). Введіть у поле «Ім'я» назву списку. У стартовому вікні майстра фільтрів IP натисніть кнопку «Додати», а потім кнопку «Далі» (Next). Виберіть один із пропонованих варіантів як джерело призначення і протокол, що відповідає вимогам завдання. Виберіть пункт «Пакети на цей порт» (To this port) і введіть у поле значення, що відповідає вимогам завдання. Для завершення роботи майстра натисніть кнопку «Готово» (Finish).

Як джерело можна вибрати:

- *«Моя IP-Адреса» (My IP Address);*
- *«Будь-яка IP-Адреса» (Any IP Address);*
- *«Певне DNS-Ім'я» (DNS name);*

– *«Певна IP-Адреса» (IP Address);*

– *«Певна підмережа IP» (IP network).*

Як призначення можна вибрати:

– *«Моя IP-Адреса» (My IP Address);*

– *«Будь-яка IP-Адреса» (Any IP Address);*

– *«Певне DNS-Ім'я» (DNS name)4*

– *«Певна IP-Адреса» (IP Address);*

– *«Певна підмережа IP» (IP network).*

Як протокол можна вибрати:

– *EGP;*

– *HMP;*

– *ICMP;*

– *RAW;*

– *RDP;*

– *RVD;*

– *TCP;*

– *UDP;*

– *XNS-IDP;*

– *інші, із вказівкою номера порту;*

– *кожний.*

3. Створіть власну дію фільтра. Для цього в діалоговому вікні «*Керування списками IP-Фільтра й діями фільтра*» (*Manage IP filter lists and filter actions*) перейдіть на закладку «*Керування діями фільтра*» (*Manage Filter Actions*) і натисніть кнопку «*Додати*». У стартовому вікні майстра налаштування дій *фільтрів* IPSec натисніть кнопку «Далі». У вікні налаштування загальних параметрів дії фільтра виберіть один із пропонованих варіантів, що відповідає вимогам завдання. Для завершення роботи майстра натисніть кнопку «*Готово*». Для завершення налаштування списку і дій фільтрів, натисніть кнопку «*Закрити*» (*Close*).

У вікні налаштування загальних параметрів дії фільтра можна вибрати одну з наступних дій:

– *«Погодити» (Negotiate);*

– *«Блокувати» (Block);*

– *«Дозволити» (Permit).*

4. Створіть свою політику *IPSec*. Для цього у вікні «*Локальні налаштування безпеки*» виберіть із контекстного меню пункт «*Створити політику безпеки IP*» (*Create IP Security Policy*) і у вікні майстра *політики IP*-безпеки натисніть кнопку «Далі». Уведіть у поле ім’я, назву вашої політики. Відмовтеся від використання пункту «*Використовувати правило за замовчуванням*» (*Activate the default response rule*) і натисніть «*Далі*». Залиште пункт «*Змінити властивості*» (*Edit properties*) і натисніть кнопку «*Готово*» для завершення роботи майстра.

5. Додайте до створеної політики правило за зазначеними критеріями. Для цього в діалоговому вікні властивостей політики натисніть кнопку «*Додати*», потім кнопку «*Далі*». З переліку списків фільтрів виберіть назву необхідної політики. Зі списку дій фільтрів виберіть одну з дій. Для завершення роботи майстра натисніть кнопку «*Готово*».

6. Установлюючи політики на сервері вашої організації, створіть для комп’ютерів, які мають використовувати безпечні з’єднання, політику, яка допускає небезпечні з’єднання з комп’ютерами, не підтримуючими *IPSec*. Передбачте, щоб комп’ютер приймав незахищений трафік, але завжди виконував спробу захистити додаткові зв’язки, посилаючи відправникові запит безпеки. Включіть в політику правило запиту безпеки для всього *IP-трафіка*, метод перевірки дійсності *Kerberos*, без параметрів тунелюваня, для типу підключення – всіх мережевих підключень.

7. Установлюючи політики на сервері вашої організації, створіть для комп’ютерів політику, яка дозволяє пересилати весь *ICMP-трафік*, не перевіряючи дійсність, без параметрів тунелюваня, для типу підключення – всіх мережевих підключень.

8. Установлюючи політики на сервері вашої організації, створіть для комп'ютерів політики відгуку за замовчуванням для відповіді на запити безпеки від інших комп'ютерів. Включіть в політику динамічний список фільтрів, для дії фільтра – відгук за замовчуванням, з методом перевірки *Kerberos,* без тунелювання, для типу підключення – всіх мережевих підключень.

9. Сервер вашої організації передає секретні дані. Установіть політики на сервері для комп'ютерів так, щоб вони завжди вимагали захист усього вихідного трафіка, допускаючи відсутність захисту тільки для початкового вхідного запиту на з'єднання. Для цього включіть правило запиту безпеки для всього *IP-трафіка,* не перевіряючи дійсність, без тунелювання, для типу підключення – всіх мережевих підключень.

10. Установіть політики на сервері вашої організації для комп'ютерів таким чином, щоб дозволити пересилати весь *ICMP-трафік*, перевіряючи дійсність методом *Kerberos,* без тунелювання, для типу підключення – всіх мережевих підключень.

11. Установіть політики на сервері вашої організації, для комп'ютерів, забезпечивши правило відгуку за замовчуванням для відповіді на запити безпеки від інших комп'ютерів. Для цього включіть в правило динамічний список фільтрів *IP*, з дією фільтра за замовчуванням, методом перевірки дійсності *Kerberos,* без тунелювання, для типу підключення – всіх мережевих підключень.

12. Створіть політики для комп'ютерів, які захищають дані за запитом, для рішення задачі, якщо клієнти інтрамережі можуть не вимагати використання *IPSec*, крім випадків, коли запит виходить із іншого комп'ютера, щоб забезпечити комп'ютеру, на якому активізована політика, відповідати на запити безпечного зв'язку. Включіть в політику правило відгуку за замовчуванням, що створює динамічні фільтри *IP* для вхідного і вихідного трафіка на основі методу перевірки дійсності *Kerberos,* без тунелювання, для типу підключення – всіх мережевих підключень.

13. Дозвольте дію, задавши у фільтрі *IP* політику для взаємодії з *DNS* сервером так, щоб дані передавалися з адреси вашого комп'ютера, на будь-яку

адресу й з будь-якої адреси на адресу вашого комп'ютера, за 53 порт *tcp* і *udp* протоколів. Включіть в політику правило зв'язку списку з дією; для всіх мережевих інтерфейсів; на основі методу перевірки дійсності *Kerberos*, без тунелювання, для типу підключення – всіх мережевих підключень.

14. Дозвольте дію, задавши у фільтрі *IP* політики для взаємодії з *SQL*-сервером так, щоб дані передавалися з адреси вашого комп'ютера на будь-яку адресу й з будь-якої адреси на адресу вашого комп'ютера, за порт *1433* порту *tcp* і *udp* протоколами. Включіть в політику правило зв'язку списку з дією; для всіх мережевих інтерфейсів; на основі методу перевірки дійсності кожної з них, без тунелювання, для типу підключення – всіх мережевих підключень.

15. Дозвольте дію, задавши у фільтрі *IP* політики для Web-сервера так, щоб з будь-якої адреси й будь-якого порту передавалися дані на адресу вашого комп'ютера, за *80*/*tcp* портом й з будь-якої адреси будь-якого порту передавалися дані на адресу вашого комп'ютера за портом *443*/*tcp*.

16. Створіть політики для комп'ютерів, створивши правило для фільтрації вихідного трафіка, тобто від адреси комп'ютера на будь-яку адресу, за портом *1434*/*udp*. За допомогою цього правила заблокуйте будь-який вихідний трафік через порти *UDP 1434* інших комп'ютерів мережі.

17. Налаштуйте політики ізолювання доменів, використовуючи такі правила. Використовуйте налаштування для всього трафіка *IP*, дія фільтра: тільки *Encapsulating Security Payload* (*ESP*), нульове шифрування (null encryption), перевірка цілісності *SHA-1* (*SHA-1 integrity*), обов'язковий захист (*require security*), заборона взаємодії між комп'ютерами без використання протоколу *IPSec*. При цьому дозволивши дії фільтрів, що містять адреси або діапазон адрес контролерів домену.

18. Призначте обрану політику. Для цього виберіть пункт «*Призначити*» (*Assign*) політики «*Назва вашої політики*».

Зміст звіту

1. Назва та мета роботи.

2. Опис виконання роботи.

3. Письмові відповіді на контрольні запитання.

Контрольні питання

1. Наведіть класифікацію загроз безпеці у віртуальному середовищі.
2. Поясніть, у чому полягає метод захищеної передачі даних з використанням IPSec?
3. Як реалізовано метод захищеної передачі даних з використанням IPSec?
4. Поясніть призначення протоколу ESP.
5. Поясніть призначення й принцип роботи транспортного й тунельного режиму роботи IPSec.
6. Опишіть функції IPSec.
7. Опишіть функціональне призначення політик, які IPSec використовує за замовчуванням.
8. Які порти використовуються при передачі за протоколом ESP?
9. Опишіть можливості фільтра IPSec.

Література: [3, 15, 17].

Лабораторна робота № 9

Тема. Інформаційна безпека у віртуальному просторі

Мета: ознайомитись з механізмами виявлення виняткових ситуацій у Web- і хмарних системах, побудованих на основі різних платформ і технологій Web-сервісів, та вразливостями програмного забезпечення, отримати практичні навички роботи з базами даних вразливостей.

Короткі теоретичні відомості

Стійкість комп’ютерних систем визначається їх здатністю виявляти і протидіяти відмовам і збоям, частина з яких не була передбачена на етапі проектування цих систем. Особливо гостро таке завдання стоїть при побудові Web- і хмарних систем, які характеризуються глобальною розподіленістю компонентів, їх гетерогенністю, високою складністю. При побудові цих систем

практично неможливо передбачити всі можливі виняткові ситуації (помилки, відмови і збої) і реалізувати методи протидії.

Іншим важливим аспектом створення і експлуатації Web- і хмарних систем є їх потенційна вразливість до кібер-атак та інформаційних вторгнень, метою яких є порушення доступності послуг, що надаються, цілісності або ж конфіденційності даних.

Концепція сервіс-орієнтованої архітектури (Service-Oriented Architecture, SOA) була запропонована для вирішення проблем забезпечення ефективної, надійної і безпечної взаємодії складних розподілених систем. SOA допускає, що сучасні Web-системи мають будуватися на основі слабо пов'язаних програмних модулів (служб), які мають загальнодоступні інтерфейси (описані за допомогою мови опису Web-служб WSDL – Web Service Description Language) і спеціальний механізм взаємодії (за допомогою протоколу SOAP – Simple Object Access Protocol). Описи цих модулів можуть бути виявлені іншими програмними системами в спеціальних реєстрах UDDI (Universal Description, Discovery, and Integration), після чого модулі можуть бути викликані за допомогою SOAP-повідомлень на основі мови XML, що передаються за допомогою стандартних інтернет-протоколів, таких як HTTP, SMTP, FTP та ін.

Досягнення високої надійності і стійкості сервіс-орієнтованої архітектури має вирішальне значення для критичних і бізнес-критичних сфер, таких як телекомунікаційні системи, системи електронної комерції, банкінгу, інтернету речей та ін. Знання точних причин і джерел виникнення виключень, що виникають під час роботи Web-служби, дозволяє розробникам застосовувати найбільш зручні методи забезпечення відмовостійкості і відновлення після помилок.

В сервіс-орієнтованих Web-системах застосовуються такі механізми відмовостійкості:

1) backward error recovery (повернення системи до попереднього безпомилкового стану);

2) forward error recovery (перехід системи в один з наступних безпомилкових станів).

Останній зазвичай залежить від логіки додатка і використовує механізми обробки винятків. Оскільки повернення до попереднього безпомилкового стану не завжди можна застосувати до Web-служб через те, що більшість з них не зберігають свого стану (тобто є stateless), обробка винятків стає найбільш популярним методом забезпечення відмовостійкості і відновлення після помилок Web-служб.

Для експериментального аналізу механізмів поширення винятків в Web-системах для виявлення відмінностей в обробці помилок і затримках поширення під час використання різних технологій реалізації Web-сервісів (наприклад, IBM WebSphere SDK, Apach Axis, Microsoft .Net та ін.) використовують методику засіву помилок/дефектів (fault injection).

Помилки, дефекти, відмови і збої Web-служб. Загальноприйнята концепція надійності комп'ютерних систем визначає такі загрози їх функціонування: помилки (errors), дефекти (faults, bugs) і відмови/збої (failures).

Дефект в комп'ютерній системі, наприклад, дефект в програмному забезпеченні, допущений внаслідок помилки програміста, може привести до відмови або збою системи. Відмова/збій системи діагностується, якщо помилковий результат поширюється за межі інтерфейсу системи. Дефект – це гіпотетична причина, яка призвела до отримання помилкового результату. Зазвичай розрізняють три групи дефектів комп'ютерних систем: дефекти проектування і розробки, фізичні несправності і помилки взаємодії.

Основними етапами взаємодії Web-сервісів є:

1) прив'язка (binding) Web-служби;

2) виклик (invocation) Web-служби;

3) обмін повідомленнями SOAP між клієнтом і Web-службою;

4) обробка Web-службою запитів клієнтів і підготовка результату.

Можна виокремити 18 різних помилок/дефектів, специфічних для сервіс-орієнтованих систем, що виникають на всіх зазначених етапах (таблиця 1). Ці

помилки можуть бути умовно розділені на три категорії:

1) збої мережі і відмови віддаленої Web-служби;

2) внутрішні помилки і збої Web-служби;

3) помилки прив’язки на стороні клієнта.

Зазначені помилки/дефекти є загальними (неспецифічні для конкретного додатка) і можуть проявлятися в будь-якій Web-службі під час її роботи.

Повідомлення про виникнення виняткових ситуацій (exception messages), що генеруються програмною системою, є проявом симптомів виникнення/активації помилки, збою або дефекту.

Таблиця 1 – Перелік помилок, дефектів і збоїв

№	Type of error / failure	Error / failure domain
1.	Network connection break-off	Network and system failures
2.	Domain Name System is down	
3.	Loss of request / response packet	
4.	Remote host unavailable	
5.	Application Server is down	
6.	Suspension of WS during transaction	Service errors and failures
7.	System run-time error	
8.	Application run-time error	
9.	Error causing user-defined exception	
10.	Error in Target Name Space	Client-side binding errors
11.	Error in Web Service name	
12.	Error in service port name	
13.	Error in service operation’s name	
14.	Output parameter type mismatch	
15.	Input parameter type mismatch	
16.	Error in name of input parameter	
17.	Mismatching of number of input params	
18.	WS style mismatching («Rpc» or «Doc»)	

До традиційних мережевих відмов і збоїв належать недоступність служби DNS або ж втрати і спотворення мережевих пакетів. Окрім того, безвідмовна робота Web-служби залежить від безвідмовного функціонування системного програмного забезпечення, такого як Web-сервер, сервер додатків і система управління базами даних.

Помилки можуть виникати і на стороні клієнта при ранньому зв'язуванні або виклику динамічного інтерфейсу (Dynamic Invocation Interface). Наприклад, «Помилка в просторі імен цілей», «Помилка в імені Web-служби» і т. д. виникають через зміни параметрів виклику і/або невідповідностей між WSDL-описом Web-служби та фактичним інтерфейсом виклику. Нарешті, збої і відмови самих Web-служб можуть бути пов'язані з програмними і системними помилками часу виконання (run-time errors), які генерують призначені для користувача або системні виключення.

Помилки часу виконання, такі як «Переповнення стека» або «Недостатньо пам'яті», призводять до винятків на рівні системи. Виняток, що виникає внаслідок виконання операції типу «Ділення на нуль», також перехоплюється і генерується на системному рівні. Однак, таку виняткову ситуацію набагато простіше зімітувати ніж, наприклад, системні виключення, пов'язані з переповненням стека.

Типовими прикладами помилок часу виконання додатків є «Невідповідність типу операнда» або «Вихід індексу за межі масиву».

Одним із способів прогнозування безпеки компонентів сервіс-орієнтованих систем є моделювання процесів виявлення і усунення вразливостей на основі статистичних даних, зібраних за певний період життєвого циклу ПЗ. Є ряд інформаційних ресурсів Інтернет, які надають інформацію про вразливості. Наприклад, база даних вразливостей NVD (www.nvd.nist.gov) дозволяє точно ідентифікувати вразливий програмний продукт і його версію та отримати інформацію про спосіб атаки, за якої дана вразливість проявляє себе, різновиди загроз та іншу корисну інформацію. База даних CVE (www.cve.mitre.org) є єдиним і первинним постачальником

ідентифікаторів вразливостей. Ці ідентифікатори використовуються для однозначного позначення однієї і тієї ж вразливості іншими відомими базами даних.

Порядок виконання роботи

1. Отримати у викладача варіант завдання.

2. Завантажити бази даних вразливостей NVD і CVE.

3. Виконати пошук вразливостей за останній рік в базі даних NVD для програмного забезпечення Web-сервера, конфігурація якого визначається варіантом завдання.

4. Проаналізувати характеристики відібраних вразливостей.

5. Використовуючи ідентифікатори CVE-вразливостей, відібраних з бази даних NVD, зробити вибірку інформації про дані вразливості по базі даних CVE.

6. Виконати аналіз дат публікації інформації про еквівалентні вразливості в CVE і NVD базах даних.

7. Побудувати кумулятивні графіки виявлення вразливостей в програмному забезпеченні з урахуванням дат їх публікації в базах даних CVE і NVD.

8. Виконати пошук інформації про підтвердження уразливості і її виправлення (вихід патча) на Web-ресурсах розробника програмного забезпечення.

9. Побудувати графіки поточної кількості вразливостей в програмному забезпеченні з урахуванням дат їх виявлення (публікації в базі CVE) і виправлення.

Зміст звіту

1. Назва та мета роботи.

2. Опис виконання роботи.

3. Письмові відповіді на контрольні запитання.

Контрольні питання

1. Поясніть відмінність між помилкою, дефектом, відмовою і збоєм.

2. Які помилки і дефекти є найбільш характерними для Web-систем і Web-сервісів? З якої причини вони виникають і які їхні наслідки?

3. У чому полягає методика засіву помилок? Як вона реалізується практично для сервіс-орієнтованих систем?

4. Що таке виняткова ситуація? Для чого використовується цей механізм?

5. Які конструкції використовуються для оповіщення про виняткові ситуації при розробці програмного забезпечення?

6. Чи дозволяє згенероване повідомлення про виняткову ситуацію однозначно діагностувати причину його виникнення?

7. Які існують відмінності в реалізації механізму виняткових ситуацій в різних технологіях Web-сервісів?

8. Що таке вразливість програмного забезпечення?

9. Які бази даних надають інформацію про вразливості? Яка інформація зберігається в цих базах?

10. Опишіть життєвий цикл уразливості.

11. За якими критеріями оцінюється ступінь критичності вразливостей в базі даних NVD?

12. Що таке CVE, CPE, CVSS, CWE?

13. Назвіть найбільш поширені види вразливостей.

14. Які існують способи захисту від вразливостей?

15. Що таке експлойт? Як наявність експлойта впливає на ступінь критичності уразливості?

Література: [16–17].

2 КРИТЕРІЇ ОЦІНЮВАННЯ ЗНАНЬ СТУДЕНТІВ

У 10-му семестрі студенти виконують 9 лабораторних робіт. Загальна кількість балів, яку отримують студенти за виконання та захист лабораторних робіт, становить 18 балів (максимально по 2 бали на кожну лабораторну роботу).

Шкала оцінювання знань студентів: національна та ECTS

<table>
<tr><td rowspan="2">Сума балів за всі види навчальної діяльності</td><td rowspan="2">Оцінка ECTS</td><td colspan="2">Оцінка за національною шкалою</td></tr>
<tr><td>Для іспиту, курсового проекту (роботи), практики</td><td>Для заліку</td></tr>
<tr><td>90–100</td><td>A</td><td>Відмінно</td><td rowspan="5">Зараховано</td></tr>
<tr><td>82–89</td><td>B</td><td rowspan="2">Добре</td></tr>
<tr><td>74–81</td><td>C</td></tr>
<tr><td>64–73</td><td>D</td><td rowspan="2">Задовільно</td></tr>
<tr><td>60–63</td><td>E</td></tr>
<tr><td>35–59</td><td>FX</td><td>Незадовільно з можливістю повторного складання</td><td>Не зараховано з можливістю повторного складання</td></tr>
<tr><td>0–34</td><td>F</td><td>Незадовільно з обов’язковим повторним вивченням навчальної дисципліни</td><td>Не зараховано з обов’язковим повторним вивченням навчальної дисципліни</td></tr>
</table>

СПИСОК ЛІТЕРАТУРИ

1. Столлингс В. Криптографическая защита сетей / В. Столлингс. – М. : Изд. дом «Вильямс», 2001.

2. Домарев В. В. Защита информации и безопасность компьютерных систем / В. В. Домарев. – К. : Диа-софт, 1999.

3. Інформаційна безпека інформаційно-комунікаційних систем. Лабораторний практикум. Частина 1 – Комплекси засобів захисту інформації від НСД : навч. посіб. / М. В. Захарченко, В. Г. Кононович, В. Й. Кільдішев, Д. В. Голев; під ред. ак. МАІ М. В. Захарченка.– Одеса : ОНАЗ ім. О. С. Попова, 2011. – 168 с.

4. Лисиченко М. Л. Методичні рекомендації щодо механізму перевірки письмових робіт на плагіат / М. Л. Лисиченко, В. І. Жила, А. В. Левкін. – Х.: ХНТУСГ, 2017. – 28 с.

5. Колегов Д. Н. Лабораторный практикум по основам анализа защищенности Web-приложений: учебное пособие / Д. Н.. Колегов. – Томск: Издательский Дом Томского государственного университета, 2014. – 59 с.

6. Троян С. О. Захист інформаційних ресурсів: навчально-методичний посібник до курсу «Захист інформаційних ресурсів» / С. О. Троян. – Умань: [б. в.], 2012. – 120 с.

7. OWASP Foundation. OWASP Testing Guide v4.0 [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/Web_Application_Penetration_Testing.

8. Heiderich M., Nava E., Heyes G., Lindsay D. Web Application Obfuscation. – ISBN-10: 1597496049.

9. McNab C. Network Security Assessment : Know Your Network, second edition. – ISBN-10:0-596-51030-6.

10. Ristic I. Bulletproof SSL and TLS: Understanding and deploying SSL/TLS and PKI to secure servers and Web applications. – ISBN-10: 1907117040.

11. Stuttard D., Pinto M. The Web Applicaion Hackers’s Handbook: Finding

and Exploiting Security Flaws. – ISBN-10: 1118026470.

12. Zalewski M. The Tangled Web: A Guide to Securing Modern Web Applications. – ISBN-10: 1593273886.

13. HTML5 Security Cheatsheet [Електронний ресурс]. – Режим доступу: https://html5sec.org.

14. Защита в виртуальной среде: чеклист угроз [Електронний ресурс]. – Режим доступу: https://habr.com/company/croc/blog/140044.

15. Тарасюк О. М. Безопасность и устойчивость Web- и облачных систем. Практикум / О. М. Тарасюк, А. В. Горбенко; под ред. В. С. Харченко. – Министерство образования и науки Украины, Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», 2017. – 40 с.

16. Інформаційна стійкість комп’ютерних технологій і мереж : навч. посіб. / А. В. Луговой, О. Г. Славко, П. П. Костенко, М. І. Гученко, М. М. Гузій. – Кременчук : Вид-во ПП Щербатих О. В., 2015. – 350 с.

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Інформаційна стійкість комп’ютерних технологій та мереж» для студентів денної форми навчання зі спеціальності 123 – «Комп’ютерна інженерія» освітнього ступеня «Магістр». Частина ІІ

Укладачі: д. т. н., проф. М. І. Гученко,
к. т. н. П. П. Костенко,
асист. Н. Л. Сохін

Відповідальний за випуск зав. кафедри «Комп’ютерні та інформаційні системи» проф. А. В. Луговой

Підп. до др. _____. Формат 60×84 1/16. Папір тип. Друк ризографія.
Ум. друк. арк. ____. Наклад _____ прим. Зам. №_____. Безкоштовно.

Видавничий відділ
Кременчуцького національного університету
імені Михайла Остроградського
вул. Першотравнева, 20, м. Кременчук, 39600