Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра

студента _____
академічної групи *125м-17-1*
спеціальності *125 Кібербезпека*
спеціалізації[1] _____
за освітньо-професійною програмою *Кібербезпека*

на тему *Аналіз захищеності веб-додатків від атак на прикладному рівні мережевої моделі OSI*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи				
розділів:				
спеціальний				
економічний				

Рецензент				

Нормоконтролер	ст. викл. Мєшков В.І.			

Дніпро
2018

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу ступеня магістра

студенту _____ **академічної групи** _____
(прізвище та ініціали) (шифр)

спеціальності ***125 Кібербезпека***
спеціалізації[1] _____
за освітньо-професійною програмою ***Кібербезпека***

на тему ***Аналіз захищеності веб-додатків від атак на прикладному рівні мережевої моделі OSI***

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від _____ № _____

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень _____

Предмет досліджень _____

Мета _____

Вихідні дані для проведення роботи _____

3 ОЧІКУВАНІ РЕЗУЛЬТАТИ

Наукова новизна _____

Практична цінність _____

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт** (початок-кінець)
Огляд джерел за темою та напрям досліджень	03.09.18-06.10.18
Методи досліджень	07.10.18-31.10.18
Результати досліджень	01.11.18-24.11.18
Виконання економічного розділу	25.11.18-04.12.18
Оформлення пояснювальної записки	05.12.18-10.12.18

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект

Соціальний ефект

7 ДОДАТКОВІ ВИМОГИ

Завдання видано _____ _____
(підпис керівника) (прізвище, ініціали)

Дата видачі: 03.09.18р.

Дата подання до екзаменаційної комісії: 14.12.18р.

Прийнято до виконання _____ _____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: с., рис., табл., додатків, джерел.

Об’єкт дослідження: захист веб – додатків на прикладному рівні.

Мета роботи: розробка методики зі зниження ризику взламу веб – додатків.

Методи дослідження: системний підхід, методи порівняння, індексний метод.

У спеціальнй частині розроблено обманну методику, яка враховує безперервність роботи веб додатків і підвищує рівень захищеності.

У роботі проведено аналіз вразливостей та методик покращення захисту, структури системи реагування на компютерні надзвичайні події, баз даних та засобів пошуку вразливостей.

В економічному розділі наведено обгрунтування запропонованої методики.

Практичне значення роботи полягає у зниженні ризиків вторгнення до системи веб – додатків. Результати здійснених у дипломній роботі досліджень можуть бути використані при захисті веб – додатків та проведенні лабораторних робіт спеціалізованих курсів спеціальності «Кібербезпека».

Наукова новизна дослідження полягає у розробці доцільно нової методики для захисту веб – додатків.

Напрямки подальших досліджень полягають у перевірці ефективності розробленої методики на практиці.

Ключові слова: МЕТОДИКА ПО ЗАБЕЗПЕЧЕННЮ ЗАХИСТУ ВЕБ – ДОДАТКА, ВЕБ – СЕРВЕР, ВРАЗЛИВІСТЬ, МЕТОДИКИ, КІБЕРБЕЗПЕКА, ЗАГРОЗА, ВЕБ – ДОДАТОК, АТАКА, ЗЛОВМИСНИК.

РЕФЕРАТ

Пояснительная записка: с., Рис., Табл., Приложений, источников.

Объект исследования: защита веб – сайта – на прикладном уровне.

Методы исследования: разработка методики по снижению ризка взлома веб – приложений.

Методы прогнозирования: системный подход, методология, индексный метод.

В специальной части разработана обманная методика, которая учитывает беспрерывность работы веб приложенийи повышает уровень защищенности.

В работе проведен анализ уязвимостей и методик по улучению защиты, структуры системы реагирования на компьютерные чрезвычайные ситуации, баз данных и поиска уязвимостей.

В экономическом разделе обоснование предложенной методики.

Практическое значение работы состоит в снижении рисков вторжения в систему веб - приложений. Результаты проведенных в дипломной работе исследований могут быть использованы при защите веб - приложений и проведении лабораторных работ специализированных курсов специальности «Кибербезопасность».

Научная новизна исследования заключается в разработке целесообразно новой методики для защиты веб - приложений.

Направления дальнейших исследований заключаются в проверке эффективности разработанной методики на практике.

Ключевые слова: МЕТОДИКА ПО ОБЕСПЕЧЕНИЮ ЗАЩИТЫ ВЕБ - ПРИЛОЖЕНИЙ, ВЕБ - СЕРВЕР, УЯЗВИМОСТЬ, МЕТОДИКИ, КИБЕРБЕЗОПАСНОСТЬ, УГРОЗА, ВЭБ - ПРИЛОЖЕНИЕ, АТАКА, ЗЛОУМЫШЛЕННИК.

ABSTRACT

Explanatory note p., pic., tables., applications, sources.

Object of research: protection of web applications at the application level.

Objective: to develop a method for reducing the risk of hacking web applications.

Research methods: system approach, comparison methods, index method.

In a special part, a fraudulent technique that takes into account the continuity of web applications and increases the level of security.

The work analyzes the vulnerabilities and methods for improving security, the structure of the system for responding to computer emergencies, databases and vulnerability search tools.

In the economic section, the proposed method is justified.

The practical importance of the work is to reduce the risk of invasion of the web application system. The results of the thesis research can be used for the protection of web applications and conducting laboratory work of specialized courses specialty "Cyber Security".

The scientific novelty of the study is to adapt the testing methods to develop a new methodology for the protection of web applications.

The directions of further research are to test the effectiveness of the developed methodology in practice.

Key words: WEB - ADDITIONAL PROTECTION METHOD, WEB SERVER, VULNERABILITY, METHODS, CYBERSECURITY, THREAT, WEB APPLICATION, ATTACK, WRECKER.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CERT (Computer Emergency Response Team) – команда реагування на комп'ютерні надзвичайні події

CERT – UA (Computer Emergency Response Team – Ukraine) – команда реагування на комп'ютерні надзвичайні події в Україні

CNA (CVE Numbering Authorities) – адміністрація з нумерації CVE

CSS (Cascading Style Sheets) – каскадні таблиці стилей CVE (Common Vulnerabilities and Exposures) – загальні вразливості та ризики

CVSS (Common Vulnerability Scoring System) – загальна система оцінки вразливостей

CWE (Common Weakness Enumeration) – загальний перелік слабких місць

FIRST (Forum of Incident Response and Security Teams) – форум команд реагування на надзвичайні події FTP (File Transfer Protocol) – протокол передачі файлів

HTML (Hypertext Markup Language) – мова гіпертекстової розмітки

HTTP (Hypertext Transfer Protocol) – протокол передачі гіпертексту

IBM (International Business Machines) – міжнародні бізнес - машини

ID (Identifier) – ідентифікатор

IEC (International Electrotechnical Commission) – міжнародна комісія з електротехніки IP (Internet Protocol) – міжмережевий протокол

ISSAF (Information Systems Security Assessment Framework) - система оцінки безпеки інформаційних систем

ISO (International Organization for Standardization) – міжнародна організація по стандартизації

LDAP (Lightweight Directory Access Protocol) – полегшений протокол доступу до директорій

NIST (National Institute of Standards and Technology) – національний інститут стандартів та технологій 10 NVD (National Vulnerability Database) – національна база даних вразливостей

OS (operating system) – операційна система

OSSIG (Open Information Systems Security Group) – група з безпеки відкритих інформаційних систем

OWASP (Open Web Application Security Project) – відкритий проект забезпечення безпеки веб-додатків

PHP (Hypertext Preprocessor) – препроцесор гіпертексту

PTES (Penetration Testing Execution Standard) – стандарт виконання тестування на проникнення

PTF (Penetration Testing Framework) – структура тестування на проникнення

RFI (Remote File Inclusion) – віддалене виконання файлів

SMTP (Simple Mail Transfer Protocol) – простий протокол передачі пошти

SQL (Structured Query Language) – мова структурованих запитів

SSL (Secure Sockets Layer) – рівень захищених сокетів

TF – CSIRT (Task Force – Collaboration Security Incident Response Teams) – робоча група з взаємодії команд реагування на надзвичайні події

TLS (Transport Layer Security) – протокол захисту транспортного рівня

URL (Uniform Resource Locator) – уніфікований локатор ресурсів

VND (Vulnerability Notes Database) – база даних записів про вразливості

WASC (Web Application Security Consortium) – Консорціум безпеки веб – додатків

XML (eXtensible Markup Language) – розширювана мова розмітки

XSS (Cross – site Scripting) – міжсайтове виконання сценаріїв

ДСТУ – державний стандарт України

НД ТЗІ – нормативний документ системи технічного захисту інформації

ОС – операційна система

ПЗ – програмне забезпечення

RCE (Remote Code Execution) – комп'ютерна вразливість, при якій відбувається віддалене виконання коду на зламаному комп'ютері, сервері і т.п

ЗМІСТ

ВСТУП

Популярність Інтернету стала повсюдною і важливою для повсякденного життя людей та бізнесу.

Кількість веб-додатків в Інтернеті швидко зростає, але, на жаль, багато з них мають дірки у безпеці. Ці вразливості можуть варіюватися від дрібних проблем до катастрофи для хост-системи та її власників. Системи піддаються нападам з різних причин, включаючи саботаж, крадіжку інформації або грошові вигоди, такі як установка спам-ботів, які потім продаються спамерам. Незалежно від того, для чого атаки створені, важливо, щоб вони не вдалися [1].

Веб-сервери та веб-додатки використовуються для обробки завдань і даних, які можуть бути критичними та дуже цінними що робить їх дуже привабливими для атак. Існуючі методи розпізнавання та профілактики не можуть впоратися з обсягом і витонченістю нових атак, що постійно з'являються, що свідчить про необхідність створення нових додаткових шарів захисту.

Хитрощі дають альтернативні шляхи для захисту систем, які можуть забезпечити додатковий шар захисту. Використання хитрощів - це не нова концепція, але це традиційно було обмежено тимчасовими підходами, реалізованими в окремих інструментах. Однак вони забезпечують корисний додатковий шар який може бути інтегрований у будь-яку виробничу систему на будь-якому шарі чутливому до атак, включаючи протоколи додатків, такі як HTTP.

Поточні обманні рішення, як правило, не торкаються протоколів прикладного рівня. Отже, ці рішення не можуть належним чином використовується для захисту від нападів на прикладному рівні.

Ця робота спрямована на вивчення того, як захистити веб додатки на прикладному рівні моделі OSI використовуючи обман атакуючих [2].

РОЗДІЛ 1
АНАЛІЗ ВРАЗЛИВОСТЕЙ ТА МЕТОДИК ЗМЕНШЕННЯ РИЗИКІВ

1.1 Аналіз принципів функціонування веб – додатків та технологій їх побудування

Веб-додаток — додаток, в якому клієнтом є оглядач Інтернета, а сервером — веб-сервер. Оглядач Інтернета може бути реалізацією так званих тонких клієнтів. Він відобрадає веб-сторінки і, як правило, входит до складу операційної системи, а його оновлення та супровід виконує постачальник оперційної системи. Логіка додатка зосереджена на сервері, а оглядач Інтернета найчастіше відповідає лише за відображення інформації, завантаженої з сервера, і за передачу на сервер даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, і веб-додатки, таким чином, є міжплатформеними сервісами [3].

Web-додатки являють собою особливий тип програм, побудованих за архітектурою "клієнт-сервер". Особливість їх полягає в тому, що саме Web-додаток знаходиться і виконується на сервері - клієнт при цьому отримує тільки результати роботи. Робота програми ґрунтується на отриманні запитів від користувача (клієнта), їх обробці та видачі результату. Передача запитів і результатів їх обробки відбувається через Інтернет Відображенням результатів запитів, а також прийомом даних від клієнта і як вони будуть передані на сервер зазвичай займається спеціальний додаток - браузер (Internet Explorer, Mozilla, Opera і т. Д.). Як відомо, однією з функцій браузера є відображення даних, отриманих з Інтернету, у вигляді сторінки, описаної на мові HTML, отже, результат, який передається сервером клієнтові, повинен бути представлений на цій мові [4].

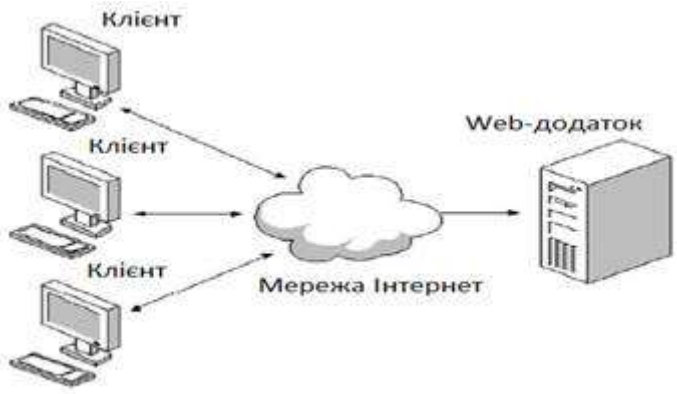

Рисунок 1.1 – Схема роботи веб-додатка

На стороні сервера Web-додаток виконується спеціальним програмним забезпеченням (Web-сервером), який і приймає запити клієнтів, обробляє їх, формує відповідь у вигляді сторінки, описаної на мові HTML, і передає його клієнту. Одним з таких Web-серверів є Internet Information Services (IIS) компанії Microsoft. Це єдиний Web-сервер, який здатний виконувати Web-додатки, створені з використанням технології ASP.NET.

В процесі обробки запиту користувача Web-додаток компонує відповідь на основі виконання програмного коду, що працює на стороні сервера, Web-форми, сторінки HTML, інший вміст, включаючи графічні файли. В результаті, як вже було сказано, формується HTML-сторінка, яка і відправляється клієнту. Виходить, що результат роботи Web-додатки ідентичний результату запиту до традиційного Web-сайту, проте, на відміну від нього, Web-додаток генерує HTML-код в залежності від запиту користувача, а не просто передає його клієнту в тому вигляді, в якому цей код зберігається в файлі на стороні сервера. Тобто Web-додаток динамічно формує відповідь за допомогою коду, - так званої виконуваної частини [5].

За рахунок наявності виконуваної частини, Web-додатки здатні виконувати практично ті ж операції, що і звичайні Windows-додатки, з тим лише обмеженням, що код виконується на сервері, в якості інтерфейсу системи виступає браузер, а в якості середовища, за допомогою якої відбувається обмін даними, - Інтернет. До найбільш типовим обробки, яка виконується Web-додатками, відносяться:

– прийом даних від користувача і збереження їх на сервері;

– виконання різних дій за запитом користувача: вилучення даних з бази даних (БД), додавання, видалення, зміна даних в БД, проведення складних обчислень;

– аутентіфіціроваліе користувача і відображення інтерфейсу системи, яке відповідає даному користувачеві;

– відображення постійно змінюється оперативної інформації [6].

1.2 Аналіз існуючих вразливостей веб – додатків

Слово security походить від латинського слова «Se-Cura», яке перекладається «без турботи» або «безтурботний», що означає що вам не потрібно ні про що піклуватися. Вам не потрібно турбуватися про постійні погрози так чи інакше, тому що різні механізми безпеки повинні захищати вас, вашу систему і ваші дані від несанкціонованого доступу, модифікації або нанесення шкоди.

Тобто захищена комп'ютерна система повинна протистояти спробам вторгнення і захищати себе від доступу неавторизованими людьми. Вона повинна подбати про обробку даних методом, який не призведе до небажаних результатів, наприклад, припинення роботи або спотворення даних. Програмісти досить часто вважають, що весь написаний код правильний, а насправді виявляється не так.

В Інтернеті існує безліч веб-додатків, які обробляють дані і запити від користувачів, які пройшли перевірку автентичності (що увійшли в систему), і від тих, хто цього не зробив.

Таблиця 1.1

Значення впливу успішної експлуатації вразливості на збереження конфіденційності

Рівень впливу на збереження конфіденційності	Опис
Високий	Повна втрата конфіденційності, всі ресурси стають відкритими для зловмисника

Продовження табл.1.1

Рівень впливу на збереження конфіденційності	Опис
Низький	Розкриття інформації не призводить до серйозного збитку від впливу на вразливий компонент.
Вплив відсутній	Втрати конфіденційності відсутні

Користувач, який не пройшов перевірку автентичності, не повинен мати можливість відправляти запити або дані в додаток, яке дає їм підвищені права, що виходять за рамки встановлених системним власником. Додаток також має бути стабільним щоб користувачі могли завжди мати до нього доступ [7].

На сьогоднішній день можна впевнено говорити про те, що Web-технології міцно увійшли в життя будь-якого бізнесу, будучи найбільш зручним способом надання інформації: комерційні і державні структури досить активно надають свої послуги, використовуючи публічні та приватні мережі для всіх видів користувачів. Переваги такого підходу очевидні - поліпшуються найбільш критичні показники бізнес-процесів: продуктивність, оперативність, доступність, вартість і т.п [8].

Що може статися, якщо безпеку ігнорується в програмному проекті? При виявленні уразливості безпеки, для різних користувачів / організацій вона матиме різні наслідки. Клієнти, що використовують дане програмне забезпечення, ймовірно, будуть найбільш порушені в найкоротші терміни. Цілком можливо, у цих клієнтів є свої клієнти, котрих торкнеться дана проблема. У довгостроковій перспективі організація, що надає або в яких беруть неефективне і небезпечне програмне забезпечення, швидше за все втратить клієнтів.

Незалежно від того, яких збитків викликаний проблемою безпеки, це те, чого ви б вважали за краще уникнути. Це буде коштувати як часу, так і грошей для аналізу і ремонту які краще будуть витрачені на щось інше [9].

1.3 Десять найбільших ризиків безпеки веб-додатків за версією OWASP(Open Web Application Security Project)

Кількість загроз зростає пропорційно зростанню бізнесу, проте як показала багаторічна практика, 99% атак відбуваються через десяток стандартних помилок валідації входять даних, або виявлені вразливості в встановлених компонентах програмного забезпечення сторонніх виробників, або банально, через недбальство системних адміністраторів, які використовують налаштування і паролі, встановлені за замовчуванням.

Класифікацією векторів атак і вразливостей займається спільнота OWASP (Open Web Application Security Project). Це міжнародна некомерційна організація, зосереджена на аналізі та поліпшення безпеки програмного забезпечення.

OWASP створив список з 10-и найбільш небезпечних векторів атак на Web-додатки, цей список отримав назву OWASP TOP-10 і в ньому зосереджені найнебезпечніші уразливості, які можуть коштувати деяким людям великих грошей, або підриву ділової репутації, аж до втрати бізнесу [10].

1.3.1 Ін'єкції даних

SQL-injection в перекладі з англійської означає впровадження SQL-коду. Це один з найбільш небезпечних і поширених способів зламу сайтів і програм, що працюють з базами даних, заснований на впровадженні в запит довільного SQL-коду [11]. Впровадження SQL, в залежності від типу СУБД, яка використовується, і умов впровадження, може дати можливість атакуючому виконати довільний запит до бази даних (наприклад, прочитати вміст будь-яких таблиць, видалити, змінити або додати дані), отримати можливість читання і / або запису локальних файлів та довільних команд на сервері, який атакується [12].

Якщо веб-сайт був змінений зловмисниками, щоб доставити зловмисне програмне забезпечення для відвідувачів, багато людей вважають що у цій ситуації у всьому винні власники сайтів і це призводить до зниження репутації сервісу.

Заробіток репутації займає велику кількість часу, але може бути зруйнований дуже швидко. Існує висока ймовірність, що веб-сайт опиниться у чорних списках, які попереджають про зловмисні сайти, навіть якщо фактичні наміри сайту не є зловмисними [13].

Запобігання ін'єкції вимагає збереження даних окремо від команд та запитів.

– Найкращим варіантом є використання безпечного API, який уникає використання перекладача цілком або забезпечує параметризований інтерфейс,або перемістити, щоб використовувати Інструменти реляційного картографування об'єктів (ORM).

– Використовуйте "білий список" для перевірки вхідних даних на серверній стороні. Це не повний захист, тому що багато додатків вимагають спеціальні символи.

– Для будь-яких залишкових динамічних запитів уникайте спеціальних символів використовуючи спеціальний синтаксис викрадення для цього інтерпретатора.

– Використовувати LIMIT та інші елементи керування SQL у запитах, щоб запобігти масове розкриття записів у разі введення SQL [14].

1.3.2. Недоліки системи аутентифікації і зберігання сесій

Для того, щоб відрізняти одного користувача від іншого, web-додаток використовує так звані сесійні куки. Після того, як Ви ввели логін і пароль і додаток вас авторизуватися, в сховище браузера зберігається спеціальний ідентифікатор, який браузер надалі пред'являє серверу при кожному запиті сторінки вашого web-додатку. Саме так web-додаток розуміє, що Ви це саме Ви [15].

У разі, якщо ваш ідентифікатор вкраде зловмисник, а в системі не були реалізовані перевірки, скажімо IP-адреси сесії, або перевірки наявності більш одного з'єднання в одній сесії, зловмисник зможе отримати доступ до системи з правами вашого облікового запису. А якщо це інтернет-банк або кабінет платіжної системи, про наслідки такого несанкціонованого доступу Ви можете легко здогадатися самі [16].

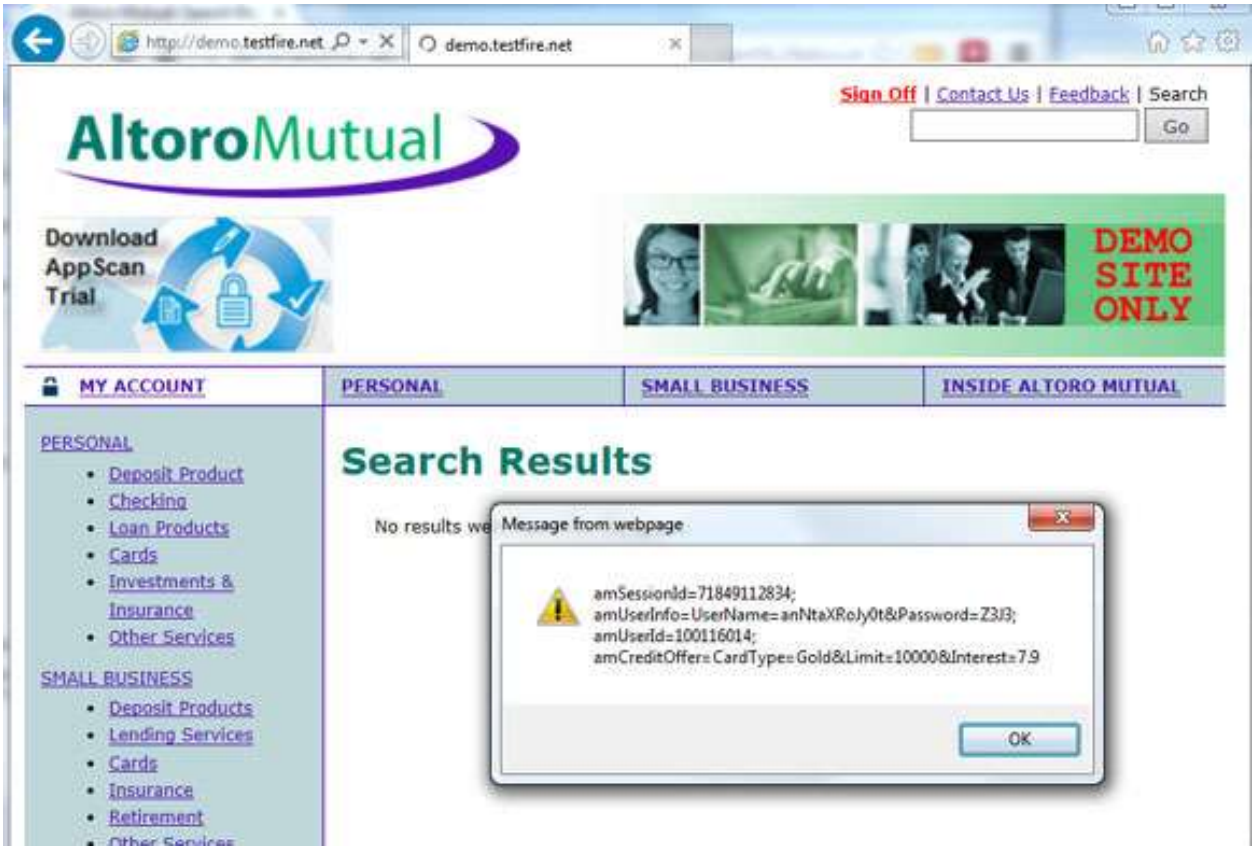

Рисунок 1.2 – Результат роботи скрипта

За можливості застосовуйте багатофакторну аутентифікацію до запобігання атакам грубої сили та атакам повторного використання облікових даних.

– Не завантажуйте та не використовуйте будь-які облікові дані за умовчанням, зокрема для адміністраторів.

– Виконуйте перевірки слабких паролів, наприклад, тестування нових або змінених паролів зі списком з 10000 найгірших паролів.

– Переконайтеся, що реєстрація, відновлення облікових даних та шляхи API

– укріплені проти атак перерахування рахунків за допомогою однакового повідомлення для всіх результатів.

– Обмеження або все більша затримка спроб входу в систему. Введіть всі помилки і сповіщати адміністраторів про накладення наказів, грубу силу або інші атаки виявляються.

– Використовуйте серверний, захищений, вбудований менеджер сесій, який генерує новий випадковий ідентифікатор сеансу з високою ентропією після логіну.

Ідентифікатори сеансу не повинні міститися в URL-адресі, надійно зберігатися і бути недійсними після виходу з системи та простою [17].

1.3.3. Незахищеність критичних даних

Багато веб-додатків не захищають конфіденційні дані, такі як кредитні карти і облікові дані для аутентифікації. Зловмисники можуть вкрасти або модифікувати такі слабо захищені дані для використання в своїх корисливих цілях [18].

Найпростіший приклад - передача даних по протоколу HTTP. Справа в тому, що дані передаються по протоколу HTTP ніяк не зашифровано, а при проходженні даних від комп'ютера користувача до Web-сервера, дані пройдуть досить багато різних вузлів: маршрутизатор офісу або домашній роутер, маршрутизатор провайдера, маршрутизатор на каналі, маршрутизатор в дата- центрі хостинг-провайдера сервера і так далі. На кожному з цих вузлів може зачаїтися так званий сніфер, програма, яка зчитує весь трафік і передає зловмисникові. А останній переглядає отримані дані на предмет персональних даних та даних кредитних карт [19].

Такі дані повинні передаватися виключно за протоколом HTTPS, про що повинен говорити відповідний напис в адресному рядку браузера:

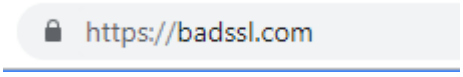

Рисунок 1.3 – Приклад сайту з протоколом HTTPS

Ще одне завдання SSL-сертифіката (а саме так називається спеціальний ключ, за допомогою якого здійснюється перевірка справжності та шифрування в HTTPS) - підтвердити, що він виданий саме для даного сайту. У разі, якщо сертифікат прострочений або підроблений, ви побачите наступну картину:

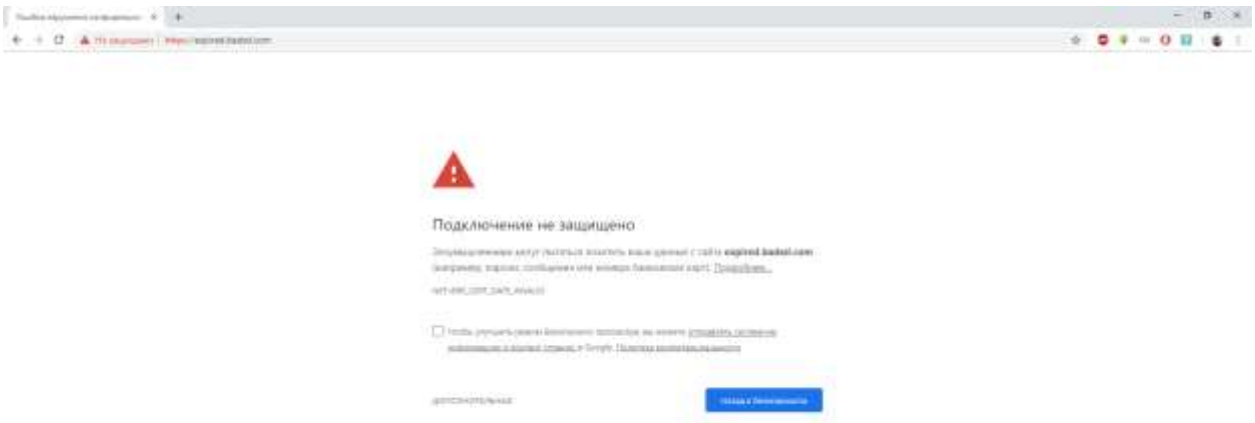

Рисунок 1.4 – Сторінка про не правильний сертифікат SSL

Інший приклад - відсутність шифрування критичних даних, таких як паролі або номери кредитних карт. У разі, якщо дані зашифровані, то навіть у разі отримання несанкціонованого доступу на сервер, зловмисник не зможе вкрасти критичні дані [20]. До паролів, зокрема, повинна застосовуватися необоротна хеш-функція - розшифрувати шифрограму при цьому не можливо і перевірка пароля відбувається шляхом формування шифрограми введеного пароля і порівняння її з наявною в базі.

– Класифікувати дані, які обробляються, зберігаються або передаються додатком. Визначте, які дані є чутливими відповідно до конфіденційності законів, нормативних вимог чи бізнес-потреб.

– Застосувати елементи керування відповідно до класифікації.

– Не зберігайте конфіденційні дані без необхідності. Відкиньте це як тільки можливо або використовувати укорочення. Дані, які не зберігаються, не можуть бути викрадені.

– Обов'язково шифруйте всі конфіденційні дані в стані спокою.

– Забезпечити найсучасніші та надійні стандартні алгоритми, протоколи, і ключі; використовувати правильне управління ключами.

– Шифрувати всі транзакції за допомогою захищених протоколів, таких як TLS з чутливими шифрами для шифрування секретності (PFS). Забезпечити шифрування використовуючи директиви типу HTTP Strict Security (HSTS).
– Вимкнути кешування для відкликів, які містять конфіденційні дані [21].

1.3.4 Експлуатація вразливостей XXE

Атака зовнішніх об'єктів XML - це тип атаки на додаток, що аналізує вхід XML. Ця атака виникає, коли XML-вхід, що містить посилання на зовнішній об'єкт, обробляється слабко налаштованим XML-аналізатором. Ця атака може призвести до розголошення конфіденційних даних, відмови в обслуговуванні, підробки запиту на стороні сервера, сканування портів з точки зору машини, де розташований аналізатор, та інших системних впливів [22].

Стандарт XML 1.0 визначає структуру XML-документа. Стандарт визначає поняття, яке називається сутність, яка є блоком зберігання деякого типу. Існує декілька різних типів сутностей, зовнішній загальний / параметр, що розбирається, часто скорочується до зовнішнього об'єкта, який може отримати доступ до локального або віддаленого вмісту через оголошений системний ідентифікатор. Ідентифікатор системи вважається URI, який може бути переадресовано процесором XML при обробці об'єкта. Процесор XML замінює випадки названого зовнішнього об'єкта з вмістом, який виділяється ідентифікатором системи. Якщо системний ідентифікатор містить зіпсовані дані та XML-процесори, які не використовують ці дані, процесор XML може розкрити конфіденційну інформацію, яка зазвичай не доступна програмі. Подібні вектори атаки застосовують використання зовнішніх DTD, зовнішніх таблиць стилів, зовнішніх схем тощо, які, коли вони включені, дозволяють подібні атаки подібних зовнішніх ресурсі в[23].

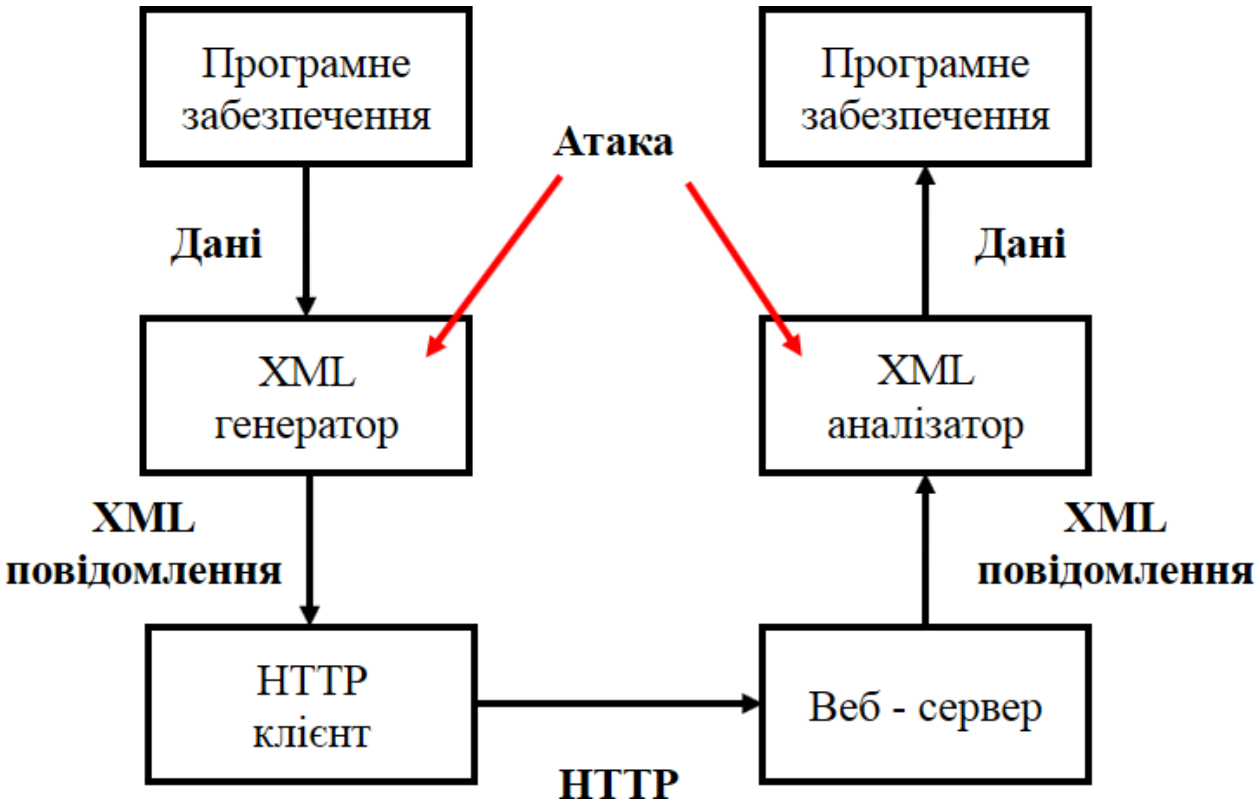

Рисунок 1.5 – Можливі місця здійснення атаки на засоби аналізу XML – вводу

Напад може включати розкриття локальних файлів, які можуть містити конфіденційні дані, такі як паролі або особисті дані користувача, за допомогою файлів. Оскільки атака відбувається відносно додатка, обробляючого XML-документ, зловмисник може використовувати цю надійну програму для обертання на інші внутрішні системи, можливо розкривши інший внутрішній вміст через HTTP-запити або запустивши атаку CSRF на незахищені внутрішні служби. У деяких ситуаціях бібліотека XML-процесорів, яка є вразливою до проблем, пов'язаних із пошкодженням пам'яті на стороні клієнта, може бути використана шляхом вилучення зловмисного URI, що дозволяє виконувати довільний код у обліковому записі програми. Інші атаки можуть отримати доступ до локальних ресурсів, які не можуть припинити повернення даних, що може вплинути на доступність додатків, якщо надто багато потоків або процесів не випущено [24].

Навчання розробників є важливим для визначення та пом'якшення XXE. Крім того, запобігання XXE вимагає:

Коли це можливо, використовуйте менш складні формати даних, такі як JSON, і уникнути серіалізації конфіденційних даних.

– Виконати оновлення всіх процесорів XML та бібліотек, що використовуються для додатку або в основній операційній системі. Використовувати перевірки залежності. Оновити SOAP до SOAP 1.2 або вище.

– Вимкнути XML-зовнішній об'єкт і обробку DTD у всіх XML-файлах аналізатори в додатку.

– Впровадити позитивну ("білу") серверну верифікацію входу.

– Переконатися, що функціональність завантаження файлів XML або XSL перевіряється, використовуючи перевірку XSD або аналогічну.

– SAST інструменти можуть допомогти виявити XXE у вихідному коді, хоча

Перевірка ручного коду є найкращою альтернативою у великому, складному додатку з багатьма інтеграціями [25].

1.3.5 Порушення контролю доступу

Контроль доступу, який іноді називається авторизацією, полягає в тому, як веб-додаток надає доступ до вмісту та функцій деяким користувачам, а не іншим. Ці перевірки виконуються після автентифікації та регулюють те, що дозволено користувачам «дозволених». Контроль доступу звучить як проста проблема, але підступно важко правильно реалізувати [26]. Модель контролю доступу веб-програми тісно пов'язана з вмістом та функціями, які надає сайт. Крім того, користувачі можуть потрапляти до кількох груп або ролей з різними можливостями або привілеями [27].

Розробники часто недооцінюють труднощі впровадження надійного механізму контролю доступу. Багато з цих схем не були свідомо розроблені, а просто розвивалися разом з веб-сайтом. У цих випадках правила контролю доступу вставляються в різні місця по всьому коду. Оскільки сайт розширюється, спеціальна збірка правил стає настільки громіздкою, що її майже неможливо зрозуміти [28].

Багато з цих неякісних схем контролю доступу нелегко виявити та використати. Часто все, що потрібно, - це оформлення запиту на функції чи вміст, які не повинні надаватися. Як тільки виявлено дефект, наслідки недосконалої схеми контролю доступу можуть бути руйнівними. На додаток до перегляду несанкціонованого вмісту, зловмисник може змінити або видалити вміст, виконувати несанкціоновані функції або навіть взяти на себе адміністрування сайту [29].

Однією із специфічних проблем із контролем доступу є адміністративні інтерфейси, які дозволяють адміністраторам сайтів керувати сайтом через Інтернет. Такі функції часто використовуються, щоб дозволити адміністраторам сайтів ефективно керувати користувачами, даними та вмістом на своєму сайті. У багатьох випадках сайти підтримують різноманітні адміністративні ролі, щоб дозволити точніше деталізувати адміністрування сайту [30]. Завдяки своїй силі ці інтерфейси часто є основними цілями для атак як сторонніх, так і внутрішніх [31].

Контроль доступу є ефективним лише в тому випадку, якщо він виконується на надійній серверній стороні, де атакуючий не може змінити перевірку контролю доступу або метадані.

– Реалізувати механізми контролю доступу один раз та повторно їх використовувати протягом усього застосування, включаючи мінімізацію використання CORS.

– Моделі контролю доступу повинні забезпечувати право власності на записи.

– Помилки контролю доступу до журналу, адміністрування сповіщень при необхідності (наприклад, повторні невдачі).

– API обмеження швидкості та доступ до контролера для мінімізації шкоди від

– автоматичних атак.

– Токени JWT повинні бути недійсними на сервері після виходу з системи.

– Розробники та персонал з питань якості повинні включати функціональний контроль доступу одиничні та інтегральні тести [32].

1.3.6 Небезпечна конфігурація

Безпека Web-додатків вимагає наявності безпечної конфігурації всіх компонентів інфраструктури: компонентів програми (таких як фреймворки - frameworks), веб-сервера, сервера баз даних і самої платформи. Налаштування компонентів сервера за замовчуванням найчастіше небезпечні і відкривають можливості до атак[33]. Наприклад, крадіжка сесійного cookie через JavaScript при XSS-атаці стає можливою завдяки виключеною за замовчуванням налаштування cookie_http only[34].

При правильному налаштуванні сервера і включеній опції cookie_httponly, отримати сесійний cookie через JavaScript неможливо, але часто ця проста і важлива настройка була відсутня в таких критично важливих місцях, як особисті кабінети платіжних систем[35].

Ще один приклад дитячої вразливості - використання налаштувань за замовчуванням в серверах баз даних, таких як Redis, Memcached і інших - закрита служба може бути доступна на публічному IP-адресу сервера, і / або використовувалися паролі, встановлені виробником за замовчуванням. Це дозволяє зловмисникові запросто читати і змінювати дані, в числі яких, нерідко бувають і сесійні cookies (які можуть бути наслідки - ми вже знаємо) і виводяться користувачам в браузер дані (що дозволяє ще й XSS-атаку застосувати)[36].

Крім того, програмне забезпечення повинно бути в актуальному стані: уразливості знаходять кожен день в самих різних програмних компонентах - операційній системі, web-серверах, серверах баз даних, поштових серверах і т.д. І навіть якщо ваш додаток правильно написано і ретельно перевіряє всі вхідні дані, і взагалі, добре захищене, це не означає що в один прекрасний момент не знайдеться вразливість у вашій ОС або Web-сервері [37].

1.3.7 Міжсайтовий скриптинг

"Міжсайтовий скриптинг" передбачає введення шкідливого коду на сервер з

входом користувача [38].

Тим часом коли атаки в один клік зосереджуються на довірі, яку сервер має до автентифікованого користувача, міжсайтовий скриптинг націлений на те, що користувач довіряє певному веб-сайту [39].

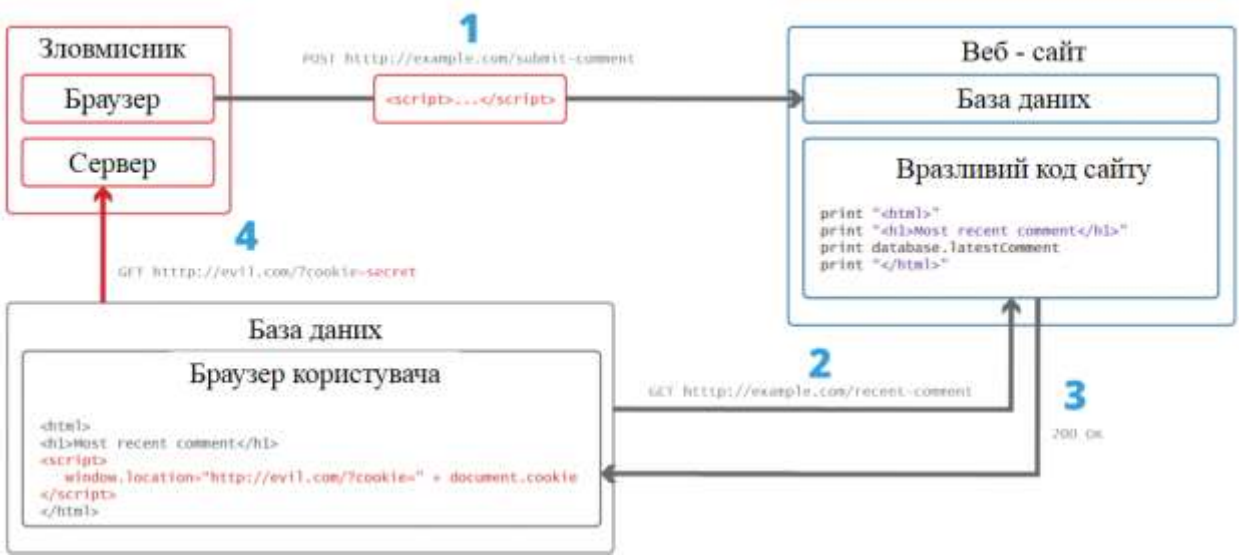

Рисунок 1.6 – Схема здійснення XSS – атаки

Специфіка подібних атак полягає в тому, що замість безпосередньої атаки сервера вони використовують вразливий сервер в якості засобу атаки на клієнта. Іноді для терміну використовують скорочення «CSS», але щоб не було плутанини з каскадними таблицями стилів, використовують скорочення «XSS». Зараз XSS складають близько 15% всіх виявлених вразливостей. Довгий час програмісти не приділяли їм належної уваги, вважаючи їх безпечними. Однак ця думка помилкова: на сторінці або в HTTP-Cookie можуть бути дуже уразливі дані (наприклад, ідентифікатор сесії адміністратора). На популярному сайті скрипт може влаштувати DoSатаку. Умовно XSS можна розділити на активні і пасивні: Пасивні Пасивні XSS передбачають, що скрипт не зберігається на сервері уразливого сайту, або ж він не може автоматично виконатися в браузері жертви. Для спрацювання пасивної XSS потрібна якась додаткова дію, яку повинен виконати браузер жертви (наприклад,

клік по спеціально сформованому посиланню). Їх також називають першим типом XSS [40].

1.3.8 Незахищена десеріалізація

Щоб зрозуміти, наскільки небезпечна десеріалізація, ми спочатку повинні зрозуміти, що таке серіалізація та десеріалізація.

Серіалізація означає процес перетворення об'єкта у формат, який може зберігатися на диску (наприклад, збережений у файлі або в храмі), надсилатись через потоки (наприклад, stdout) або надсилати через мережу. Формат, в який об'єкт серіалізований, може бути або бінарним, або структурованим текстом (наприклад, XML, JSON YAML ...). JSON і XML - це два з найбільш часто використовуваних форматів серіалізації у веб-програмах [41].

З іншого боку, десериализація - це протилежність серіалізації, тобто перетворення серійних даних, що надходять з файлу, потоку або мережевого сокета в об'єкт [42].

Веб-застосунки регулярно використовують серіалізацію та десеріалізацію, а більшість мов програмування навіть надають вбудовані можливості для серіалізації даних (особливо в загальних форматах, таких як JSON та XML). Важливо розуміти, що безпечна десеріалізація об'єктів - це звичайна практика розробки програмного забезпечення. Проблема, однак, починається при десеріалізації ненадійного введення користувача [43].

Більшість мов програмування дають можливість налаштовувати процеси десеріалізації. На жаль, зловмисник часто може зловживати цими функціями десеріалізації, якщо програма десеріалізує ненадійні дані, які контролює злочинцем. Успішні атаки з необережної десеріалізації можуть дозволити зловмиснику здійснювати атаки "відмови в обслуговуванні" (DoS), обхід аутентифікації та атаки типу RCE [44].

1.3.9 Неперевірені переадресації та пересилання

Web-додатки часто переадресовують користувача з однієї сторінки на іншу. В процесі можуть використовуватися неналежним чином перевіряються параметри із зазначенням сторінки кінцевого призначення переадресації.

Без відповідних перевірок, атакуючий може використовувати такі сторінки для переадресації жертви на підроблений сайт, який, наприклад, може мати дуже схожий або не відрізняється інтерфейс, але вкраде ваші дані кредитної картки або інші критичні конфіденційні дані [45].

Цей вид вразливостей, також як і багато інших перераховані вище, є різновидом помилок перевірки вхідних даних (input validation) [46].

1.3.10 Небезпечні прямі посилання на об'єкти

Даний вид уразливості є також наслідком недостатньої перевірки призначених для користувача даних. Суть її полягає в тому, що при виведенні будь-яких конфіденційних даних, наприклад особистих повідомлень або облікових карток клієнтів, для доступу до об'єкта використовується ідентифікатор, який передається у відкритому вигляді в адресному рядку браузера, І не реалізована перевірка прав доступу до об'єктів. Наприклад, є сторінка, яка відображає приватне повідомлення і вона має адресу виду: mysite.ru/read_message.jsp?id=123654 [47].

Перебираючи число після "id =" можна буде читати чужі приватні повідомлення. Експлуатація даної уразливості дуже проста і не вимагає взагалі ніяких спеціальних навичок - достатньо лише перебирати число в адресному рядку браузера і насолоджуватися результатом. Як ні парадоксально, але цієї дитячої хвороби, часом були схильні до досить великі європейські платіжні системи [48].

1.4 Інші вразливості

Наслідки відмови в обслуговуванні

Відмова в обслуговуванні - метод атаки для запобігання постійного обслуговування наприклад, відправивши більшу кількість запитів, ніж може обробляти система. Від цього методу атаки важко захиститися, і безпосереднім наслідком атаки є те, що ваші користувачі не зможуть використовувати конкретну послугу. Якщо послуга має вирішальне значення для організації або навіть підприємства, вартість такого порушення може виявитися високою [49].

Наслідки втрати даних

Якщо будь-яка атака призводить до втрати даних, або шляхом запуску помилки, яка буде порушувати дані або за допомогою націлених дій зі знищення даних, це може викликати необхідність роботи по відновленню втрачених даних, якщо можливо, що потребує часу і грошей. Якщо не налаштоване належне і своєчасне резервне копіювання, втрата даних може бути незворотною [50].

Наслідки крадіжки даних

Деякі атаки виконуються з наміром викрасти інформацію, яка може включати в себе списки номерів кредитних карток або іншу інформацію, що відповідає за безпеку виявленого сайту. В 2011 р. Мережа Sony PlayStation була підвергнута вторгненню, де за першими звітами мільйони номерів кредитних карток, можливо, були вкрадені. В цілому це викликало б величезну кількість проблем і величезні витрати для всіх, але крадіжка не була цілком успішною немає підтверджених доказів крадіжки. Sony нараховали 250 000 доларів США штрафу за їх неспроможність захистити дані клієнтів [51].

Наслідки вкрадення акаунтів

Залежно від того, як працює веб-сайт і як здійснюється авторизація, облікові записи користувачів можуть бути викрадені зловмисниками. Нові програми можуть бути завантажені на сайт, використовуючи його в якості кроку для нового нападу на інших. Спам це великий бізнес в наші дні, тому, встановлений спам-бот або обліковий запис буде використовуватися іншими способами для відправлення великої кількості небажаної пошти [52].

Атаки типу MiTB базуються на використанні спеціальної програми-трояна, яка виступає в ролі посередника. Така програма втручається в обмін даними між

вами та веб-сервером, як правило, під час здійснення фінансових операцій або покупок в Інтернет-магазинах [53]. Зловмисники можуть використовувати троян-посередник, тобто клавитурний шпион, програму-руткит, шкідливий вхідний об'єкт браузера або підключений модуль, для викрадення ваших даних входу в інформаційну систему банку, зміни суми трансакції або виконання додаткових транзакцій від вашого імені. Обычно такі дії здійснюються під час роботи з системою банку або інтернет-магазину[54].

Це дає змогу змінювати веб-транзакції в режимі реального часу. Веб форма може виглядати щось на зразок:

1.7 – Змінена форма

MITB-атаки складніші для виявлення оскільки це відбувається між механізмами захисту користувача та його браузером [55].

Представте собі, скільки шкоди може нанести вразливий, який пробрався всередину вашого браузера, і переглядає або змінює ті дані, які ви вводите самі або отримуєте з різних веб-сайтів. На жаль, така ситуація можлива не тільки в нашому уявленні [56].

Найбільш поширеною протидією до MITB-атак сьогодні є проведення зовнішньої перевірки, коли користувач отримує підтвердження транзакції за допомогою різного типу зв'язку, окрім того який було використано, наприклад, за допомогою телефонного дзвінка або електронної пошти [57].

Веб-атаки повторного відтворення

Веб-атаки повторного відтворення - це метод використання захопленого пакету або пакетів та повторне їх надсилання що може призвести до несподіваної або небажаної поведінки зі сторони сервера . Якщо сервер не виявляє повторно використовувані дані і приймає багаторазово передані пакети - атака успішна. Якщо

зловмисник скористується даними жертви, які генеруються JavaScript, він може увійти як жертва без жодних особливостей. Дані збираються або шляхом прослуховування трафіку або встановлюючи зловмисне програмне забезпечення на комп'ютері жертв. Звичайно, атаки на веб-відтворення запобігаються за допомогою певного маркеру сеансу кожен різний зв'язок, наприклад хешування різними ключами або використанням міток часу [58].

Використання захищених каналів, таких як SSL або TLS, також рекомендується, оскільки це ускладнює доступ до інформації, необхідної для виконання подібних атак.

Інші загрози

Є й інші загрози, які слід враховувати. "Людина в середині атаки" виникає, коли хтось підслуховуйте веб-трафік, обдурюючи інші сторони, щоб з'єднатися з атакуючим [59].

Людина-в-середній атаки перехоплює зв'язок між двома системами. Наприклад, у http-транзакції ціллю є TCP-зв'язок між клієнтом і сервером. Використовуючи різні методи, зловмисник розбиває оригінальне TCP-з'єднання на два нових з'єднання, один між клієнтом і зловмисником, а інший - між атакуючим і сервером, як показано на рисунку 1. Після перехоплення TCP-з'єднання зловмисник діє як проксі, що вміє читати, вставляти та змінювати дані в перехопленому повідомленні [60].

Напад MITM дуже ефективний через характер http-протоколу та передачі даних, які засновані на ASCII. Таким чином, можна переглядати та інтерв'ювати в протоколі http, а також у переданих даних. Так, наприклад, можна захопити сеансові файли cookie, які читають заголовок http, але також можна змінювати транзакцію грошових коштів у контексті програми [61].

Атаку MITM можна здійснити і через https-з'єднання, використовуючи ту саму техніку; Єдина різниця полягає у створенні двох незалежних сесій SSL, по одному по кожному TCP-з'єднанню. Браузер встановлює зв'язок SSL з атакуючим, а атакуючий встановлює ще одне SSL-з'єднання з веб-сервером. Загалом браузер попереджає користувача, що використовуваний цифровий сертифікат недійсний, але

користувач може ігнорувати це попередження, оскільки він не розуміє цю загрозу [62]. У деяких конкретних контекстах можливо, що попередження не з'являється, наприклад, коли зловмисник скомпрометував сертифікат Сервера або сертифікат нападника був підписаний надійною ЦС, а CN - однаковий з оригінального веб-сайту.

MITM - це не тільки техніка нападу, але також зазвичай використовується на етапі розробки веб-програми або як і раніше використовується для оцінок вразливості в Інтернеті [63].

Одним з найпоширеніших способів боротьби з атаками MITM є використання безпечного каналу такого як SSL, коли конфіденційні дані передаються між клієнтом і сервером. SSL верифікує сервера, використовуючи сертифікати власника.

«Повний перебір» означає підбір усіх можливих варіантів, поки не буде знайдено вірний. У цьому випадку змінюється XML документ за частинах до моменту знахідки збігу [64].

Метод грубої сили часто називають взлом грубою силою. Дійсно, груба сила - в даному випадку це обчислювальна потужність - яка використовується для спроби зламати код. Замість того, щоб використовувати складний алгоритм, атака грубої сили використовує скрипт або бот для надсилання догадок, доки він не потрапляє на комбінацію, яка працює.

Існує безліч інструментів, які легко доступні, щоб допомогти хакерам запустити повний перебір. Але навіть написання сценарію з нуля не було би надто складною працею. Хоча ці атаки легко виконуються, залежно від довжини та характеру пароля та обчислювальної потужності, вони можуть зайняти кілька днів, тижнів або навіть років [65].

Гібридні атаки грубої сили

Атака грубої сили використовуюча систематичний підхід до вгадування, що не використовує зовнішню логіку. Подібні атаки включають в себе словникову атаку, яка може використовувати список слів із словника, щоб зламати код. Інші атаки можуть починатися з часто використовуваних паролів. Іноді вони описуються як атаки грубої сили. Однак, оскільки вони використовують певну логіку, щоб

вирішити, які ітерації можуть бути найбільш ймовірними в першу чергу, вони більш точно називаються гібридними атаками грубої сили [66].

Зворотна атака грубої сили

Зворотна атака грубої сили передбачає використання загального пароля або групи паролів проти численних можливих імен користувачів. Ця атака не націлена на одного користувача, а може використовуватися для спроби отримати доступ до певної мережі [67].

Найкращим захистом від такого типу атак - є використання сильних паролів.

Накладення облікових записів

Накладення облікових записів це унікальна форма атаки грубої сили, яка використовує зламані паролі та імена користувача. Зможе використовувати їх для спроби отримати доступ до кількох сайтів. Після того, як вони потрапляють до облікового запису користувача, вони повністю контролюють цей обліковий запис і отримують доступ до будь-якої деталі, яку він має.

Щоб застережити подібним нападам, як правило, дозволяють лише певну кількість спроб входу або виклик затримки після невдалої спроби входу або просто заблокувати користувача після багатьох невдалих спроб [68].

1.5 Аналіз існуючих методів захисту веб додатків

Нові ризики безпеки пов'язані з перевагами розгортання веб-додатків. Для ефективного вирішення цих ризиків різноманітний контроль безпеки повинен розглядатися протягом усього життєвого циклу розробки проекту.

Щоб зрозуміти, на якому етапі життєвого циклу рекомендується контроль безпеки, я розлянув етап життєвого циклу поетапно і вкажу ключові проблеми безпеки, які потребують особливої уваги [69].

Етап опрацювання вимог

На цьому етапі команда розробників додатків повинна об'єднати всю систему і специфікації безпеки, які вимагаються різними сторонами, що беруть участь у проекті. Системні вимоги повинні надати команді розробників огляд основної мети

додатку, в тому числі те, що має робити додаток і що він не повинен робити. Ця інформація допоможе команді розробників визначити ключові елементи контролю безпеки для програми. Крім того, потрібно ввести певні елементи керування чи механізми безпеки додатку, які б відповідали правилам чи вимогам. Наприклад,

Стандарт безпеки даних для платіжних карток (PCI DSS), вимога 6, що має назву "Розробка та підтримка безпечних систем та додатків", зосереджена на створенні елементів керування, які мінімізують існування вразливостей системи та програмного забезпечення [70].

Вона визначає вимоги до безпечного використання програмного забезпечення та захисту від атак. Правильне встановлення вимог системи та вимог безпеки користувачів буде мати вирішальне значення для етапів проектування, розробки та випробувань, оскільки це підвищить загальну безпеку веб-додатка та задоволеність користувача кінцевим результатом [71].

Етап проектування

Стадія проектування передбачає не тільки оформлення веб додатку відповідно до специфікації, викладеної на першому етапі, але також визначення стандартів безпеки кодування, виконується моделювання загрози та розробляється архітектура безпеки для додатків.

Визначення стандартів безпеки кодування

Стандарти безпеки кодування це рекомендації щодо того, як розробники повинні писати код і має містити вказівки щодо розробки безпечного коду та належних коментаріїв, щоб визначити сфери високого ризику.

Перевірте всі вхідні параметри, щоб запобігти атакам типу ін'єкцій даних та міжсайтовому скриптингу:

Програмісти повинні розробити централізований модуль для виконання вхідних параметрів з перевіркою кожного вхідного параметру, яка визначає, які типи вхідних даних буде дозволено. Спеціальні символи, такі як "~! # $% ^ & * [] <>" \ R \ n ", що надходить з форми введення, слід відфільтрувати або замінити.

Додаток повинен приймати лише дані, що містять суворо обмежені та очікувані дані, якщо очікується число, слід приймати лише цифри. Якщо слово, слід

дозволити лише букви. Вхідні дані також повинні бути перевірені на правильність формату, якщо очікується електронна адреса, лише букви, цифри, "собачка" (@) символ, дефіс та крапки в правильному розташуванні можуть бути прийняті.

Обмеження мінімальної та максимальної довжини всіх вхідних даних також повинно перевірятися. Цей метод слід використовувати для номерів облікових записів, облікових даних сеансу, імен користувачів і так далі. Всі ці методи обмежують кількість потенційних атак [72].

Оброблена відповідь додатка

Для здійснення будь-якої обробки необхідно розробити централізований модуль. Усі вихідні дані, коди повернення та коди помилок від відкликів (наприклад, виклики до бази даних) повинні перевірятися. Наприклад, непотрібна внутрішня інформація, така як IP-адреса, внутрішні імена хостів, внутрішні структури каталогів, докладні повідомлення про помилки, створені внутрішні помилки сервера під час відповіді не повинні відібражатися на стороні клієнта. Більшість прикладних програм/веб-серверів дозволяють створювати спеціальну сторінку помилок де відібражаються внутрішні помилка сервера.

Проблеми HTTP-довіри

Програмісти не повинні довіряти або спиратися на заголовки HTTP REFERER, оскільки цей тип даних може бути підроблен. Хіба що сильні криптографічні методи використовуються для перевірки цілісності заголовків HTTP, не можна довіряти цим параметрам, якщо вони надходять від браузера клієнта. Крім того, не припустимо, щоб приховані параметри могли бути змінені зловмисниками.

Тримайте чутливу інформацію сеансу на сервері, щоб запобігти модифікації на стороні клієнта

Не розміщуйте конфіденційну інформацію в будь-яких клієнтських файлах cookie. Якщо чутливі значення повинні бути збережені в браузері клієнта, потрібно використовувати криптографічні методи для захисту конфіденційності та цілісності даних.

Шифруйте сторінки, що містять конфіденційну інформацію, та запобігайте кешування

Сторінки, що містять конфіденційну інформацію, повинні шифруватися під час передачі належним чином алгоритмами та ключами, такими як SSL та TLS. Потрібно використовувати ActiveX для отримання та відображення конфіденційної інформації та встановлювати відповідні атрибути HTTP-заголовка для запобігання кешування, за допомогою браузера або проксі-сервера [73].

Управління сесіями

Ідентифікатор сеансу повинен бути довгим, складним, містити випадкові числа непередбачуваний, і його слід часто змінювати під час сеансу, щоб зменшити тривалість ідентифікатора сесії залишається дійсною. Крім того, ідентифікатор сеансу не повинен бути збережені в URL-адресі, постійні файли cookie, приховані поля HTML або заголовки HTTP.

Програмісти можуть розглянути можливість зберігання ідентифікаторів сеансу в сеансі веб-переглядача клієнта використовуючи SSL або TLS, ідентифікатори сеансів можуть бути захищені. Функція відключення для програми та час очікування сеансу простою слід також реалізувати. При відключенні користувача або вимкненні термінів простою сеанс не тільки повинен очищати файли cookie на стороні клієнта (якщо можливо), але також стан сеансу стороннього сервера для цього браузера плюс підключення до бекенда.

Обмеження доступу

Переконайтеся, що обліковий запис кінцевого користувача має лише певні права доступу до них функції, якими вони мають дозвіл на доступ, обмежуючи доступ до бекенда бази даних або для запуску команд SQL та команд ОС. При поданні заявки

роблять системні виклики для доступу до певних програм,а не здійснюють дзвінки до фактичного файлу назви та шляхи каталогу. Якщо хакери мають доступ до вихідного коду, вони можуть виявити інформацію на рівні системи. Використовуйте відображення, надане веб-сервером, як фільтруючий шар

Створити централізований модуль для аудиту та звітування додатків.

Використовувати найбільш підходящий тип методу автентифікації, ідентифікації та перевіряти вхідні запити користувачів [74].

1.6 Висновки до першого розділу

Захищеність веб – додатків від атак зловмисників залежить від технологій та компонентів, які використовуються при побудові веб – додатків, а також від можливих вразливостей у цих компонентах. Існують різні класифікації вразливостей, кожна атака через вразливість має свої особливості, але причина виникнення вразливостей – помилки при проектуванні, реалізації та застосуванні компонентів веб – додатків, отже виникає необхідність пошуку вразливостей та реагування на інформацію про випадки їх знайдення. Як в Україні, так і в інших країнах світу організовуються команди реагування на надзвичайні події у кібербезпеці, що складаються з експертів та дослідників, існує багато програм по отриманню винагороди за знайдені вразливості, такі як Bug Bounty. Статистичну інформацію про знайдені вразливості та їх особливості можна знайти у спеціально створених базах даних вразливостей, окремі міжнародні стандарти регулюють процес розкриття вразливостей.

Широкий вибір засобів дозволяє проводити пошук вразливостей та їх нейтралізацію, проте ефективність їх використання залежить від алгоритму дій. Методики з забезпечення захисту потребують спеціальних вміннь, великих витрат або багато часу на впровадження. Тому виникає необхідність у розробці такої методики з захисту, яка з найменших витрат ресурсів змогла забезпечити достатній рівень захисту веб додатків.

РОЗДІЛ 2
ПРОЕКТ ПІДГОТОВКИ ОБМАННОЇ СТРАТЕГІЇ ДЛЯ HTTP

2.1 Розробка обманної стратегії

Дизайн стратегії обману складається з п'яти незалежних стратегій. Кожна зі тратегій враховують різні типи атак та використовують різні елементи HTTP-протоколу для застосування обману.

Мета полягала в тому, щоб створити якнайбільшу кількість стратегій, використовуючи різні елементи HTTP протокол, який міг би включати до неї оманливі елементи.

2.1.1 Обманні коментарі

Метою цієї стратегії є виявлення спроб втручання шляхом введення в оману вмістом коментарів HTML, який може бути привабливим для зловмисника (наприклад: підроблені секретні посилання або фальшивий витік повноваження). Основна перевага цієї стратегії полягає в її простоті та відсутності ризиків перешкоджання регулярному використанню програми користувачами.

Метою цієї стратегії є виявлення спроб втручання зловмисника та моніторинг його діяльністі та намірів. Мета буде досягнута шляхом введення підроблених зв'язків у коментаріях HTML в вихідний потік серверних відповідей.

```
1  <body>
2      <!--Fake URL-->
3      <table cellpadding="0" cellspacing="0" border="0" width="100%"
       style="background: #f5f5f5; min-width: 340px; font-size: 1px; line-height:
       normal;"> ··· </table>
243 </body>
244
```

Рисунок 2.1 – Приклад обманних коментарів HTML

Зловмисник повинен вірити, що таке фальшиве посилання справжнє і випадково витікає всередині коментарів HTML. Нападник повинен Вважати, що посилання можна використовувати для доступу до високо привілейованого прихованого вмісту і повинен надіслати запит до нього.

Зловмисник буде спостерігатися якомога довше.

Для цього розглянуто два сценарії:

– Продовження: запит на фальшиві посилання поверне дійсну HTTP-відповідь яка містить нову підробку посилання, щоб зацікавити атакуючого і підштовхнути до взаємодії з системою в той час як виконується його моніторинг.

– Блокування: коли виявляється запит на фальшивий зв'язок, нападник буде позначений і оригінальна відповідь буде заблокована. Замість цього – буде повернена спеціальна сторінка симулююча несправність сайту на певний період. Нападник, як очікується, зупинить напади в цей момент і спробуйте знову підключитись після зазначеного періоду.

Обман буде інтегрований у відповідь системи шляхом виготовлення нового HTML-вмісту, що містить фальшиві посилання, які будуть діяти як приманки.

Ця стратегія не має пов'язаних ризиків, тому що вводячий в оману вміст вставляється всередині коментарів HTML які ігноруються веб-браузерами, що не створює перешкод для звичайного потоку програми.

Стратегія обману буде інтегрована в HTTP-трафік прикладного рівня. Для реалізації стратегії доведеться виконати наступні кроки:

Перехоплення HTTP-відповідей оригінальної програми та введення фальшивого коментаря до HTML. Коментар буде містити привабливу фразу та випадкову URL-адресу.

Перехоплювати HTTP-запити та перевірити, чи запит орієнтований на фальшиву URL-адресу, яку було введено у внутрішні коментарі у попередньому запиті. Якщо запит на фальшиву URL-адресу буде виявлено, система будеповинна:

Запустити сигнал виявлення вторгнення, збільшити лічильник оповіщень і ввести дані запиту в файл журналу

Повернути відповіді залежно від налаштованого режиму:

Режим продовження: повертає дійсну HTTP-відповідь з 200 кодом стану, щонайменше містить нову додаткову фальшиву URL-адресу, доступну атакувальнику.

Режим блокування: повернення дійсної відповіді HTTP, що містить спеціальну сторінку відповіді.

2.1.2 Обманні параметри запиту

Параметри вторгнення атаки намагаються зловмисно управляти параметрами запиту HTTP і спрямовуються на зміну оригінальної поведінки програми та обхід її легітимної логіки. Ця стратегія спрямована на виявлення спроб втручання в параметри, вводячи фальшиві параметри та моніторинг їх змін.

Основною метою цієї стратегії є виявлення спроб втручання зловмисника в систему моніторинг його діяльності та намірів. Мета буде досягнута шляхом введення підроблених параметрів в існуючий трафік HTTP-прикладного рівня та реагують, коли зазнають будь-яких змін.

```
array = {
"admin = false",
"Development = false",
"authorization = no",
"fullView = no",
"query = none"
}
```

Рисунок 2.2 – Приклад обманних параметрів

Нападник повинен вірити в те, що підроблені параметри є справжніми і дійсно належать до справжніх додатків і спробує змінити їх шкідливим чином.

Нападника слід контролювати стільки скільки можливо.

Для того, щоб це зробити, змінені підроблені параметри, повинні мати вплив у HTTP-відповідь, для підтримки у атакуючого зацікавленість для ведення моніторингу атакуючого.

Для цього розглянуто два сценарії:

– Продовження: запити з модифікованими підробними параметрами викликають тривогу втручання, але вихідна відповідь сервера буде і надалі повертатись користувачеві, коментар з випадковою інформацією буде введено в відповідь щоб зацікавити атакуючого до взаемодії зсистемою в той час як він буде спостерігатися.

– Блокування: запити з модифікованими підробленими параметрами викликають тривогу вторгнення, але замість повернення оригінальної відповіді сервера, буде повернута особлива сторінка, що імітує, що Веб-сайт знаходиться на обслуговуванні.

Обман буде інтегрований у відповідь системи шляхом виготовлення нових параметрів, які будуть діяти як обманка і імітує наслідки змін за цими параметрами.

Виготовлені обманливі параметри завжди повинні відрізнятись від реальних параметрів, які є реальними які програма використовує для використання, щоб не створювати перешкод для звичайної роботи програми.

Стратегія обману буде інтегрована в HTTP-трафік прикладного рівня. Для реалізації стратегії доведеться виконати наступні кроки:

Перехоплення HTTP-відповідей оригінальної програми та введення фальшивих параметрів раніше доставки відповіді клієнту. Можливі підроблені значення параметрів повинні мати привабливі імена та значення. Список кандидатів складатиметься з {"admin = false", "Development = false", "authorization = no", "fullView = no", "query = none"}.

Для запитів GET параметри будуть додані до існуючих значень посилань <a href>.

Для запитів POST параметри будуть додані як приховані параметри входу існуючих форм.

Перехоплювати HTTP-запити та перевіряти, чи введені параметри з попередніх кроків були модифіковані. Якщо виявлена модифікація, система повинна:

Запустити сигнал виявлення вторгнення, збільшити лічильник оповіщень і ввести дані запиту в файл журналу

Повернення відповіді залежно від налаштованого режиму:

Режим Continue: змінити відповідь HTTP, вставляючи коментар із випадковим вмістом і зміна підроблених параметрів на інше значення.

Режим блокування: повертає відповідь HTTP, що містить спеціальну сторінку відповіді, в якій зазначено, що система знаходиться на обслуговуванні та містить детальну інформацію про час відновлення роботи. Флаг режиму блокування буде автоматично скинуто після налаштовування час (за умовчанням = 5 хвилин).

2.1.3 Обманні сеансові файли cookie

Багато методів нападу на керування сесіями базуються на маніпуляціях або крадіжці маркерів ідентифікації сеансу HTTP, які можуть застосовуватися для обходу механізмів автентифікації веб-додатків. Ці токени зазвичай встановлюються за допомогою HTTP файлів cookie, які по суті є невеликими фрагментами даних, надісланих веб-серверами і зберігаються на клієнтському комп'ютері або веб-переглядачі.

Cookie-ідентифікатори сеансу можуть бути скомпрометовані різними способами. Одним з підходів є підслуховування вмісту файлів cookie, якщо вони передаються з використанням незахищеного підключення (HTTP замість HTTPS).

Інший підхід, використовуваний зловмисниками, базується на аналізі сеансу.

Якщо зловмисник зможе зрозуміти процес генерації ідентифікатору сеансу, він сможе , передбачати або виготовляти дійсні ідентифікатори сеансу для захоплення існуючих сеансів і обхід процесу автентифікації додатка.

Дизайн наступної стратегії обману зосереджений на тих атаках, де нападники намагаються зрозуміти і відтворити процес створення імені сесії веб-додатка.

Обманна стратегія буде імітувати слабкі шаблони створення ідентифікатора сесії на сервері, з метою звернути увагу зловмисника та переконати його чинити напад за допомогою перехоплених підробних ідентифікаторів

Метою цієї стратегії є виявлення спроб маніпулювання сесією зловмисника і дізнатися про рівень навичок, які має зловмисник. Цю мету буде досягнуто шляхом введення до файлів cookie ідентифікаційного коду підробки сеансу в існуючий трафік HTTP-прикладного рівня та реагують, коли зазнають будь-яких змін.

Зловмисник повинен повірити, що підробні ідентифікатор сеансу і Кукі-файли справжні і думаюти, що вони були створений з використанням певного шаблону, який може бути підроблений або використаний. Нападник повинен намагатися внести зміни до цих файлів cookie, щоб отримати доступ до обмежених ресурсів або видати себе за дійсні ідентифікатори сеансу.

Для цього розглянуто два сценарії:

– Продовження: запити з зміненими файлами cookie запускають сповіщення про вторгнення, але нові запити будуть продовжуватись повертаючи відповідну оригінальну відповідь і випадковий вміст всередині коментарів для того, щоб тримати зацікавленість нападаючого як можна довше.

– Блокування: При виявленнв спроби маніпулювання файлами cookie, зловмисник буде позначений і оригінальна відповідь буде заблокована. Замість цього, буде повернута спеціальна сторінка, що імітує, що веб-сайт знаходиться на обслуговуванні.

Виготовлені обманні файли cookie повинні відрізнятись від справжніх файлів cookie, які програма використовує для справжніх випадків використання, щоб не створювати перешкод для звичайного потоку програми.

Обманна стратегія буде інтегрована в HTTP-трафік прикладного рівня.

2.1.4 Обманний JavaScript

Ця стратегія використовує JavaScript як приманку для привернення уваги зловмисника з фальшивими змінними і URL-адресами. Метою цієї стратегії є виявлення зловмисників з високими атакуючими можливостями і заблокувати їх.

Зловмисник повинен проаналізувати код JavaScript, знайти змінну з привабливою підробленою URL-адресою та подати запит на цю URL-адресу.

Зловмисника слід контролювати або блокувати, залежно від налаштованої опції. Розглянемо обидва варіанти:

– Продовження: запити на URL, витягнуті з оманливого JavaScript поверне дійсну HTTP відповідь яка містить нову підробку JavaScript для збереження нападника Зацікавленим для ведення його моніторингу.

– Блокування: коли витягується запит на фальшивий зв'язок з JavaScript, зловмисника буде позначено і спеціальна сторінка буде повернута імітуючи те, що Веб-сайт знаходиться на обслуговуванні.

Обман буде інтегрований у відповідь системи, ввівши фальшиву функцію JavaScript в справжню HTTP-відповідь.

Назва файлу, що містить оманливу функцію JavaScript, завжди повинна відрізнятись від існуючих функцій JavaScript що програма використовує для справжніх випадків використання, щоб не створювати перешкод для звичайного потоку програми.

Функція, яка реалізує стратегію, повинна виконати наступні кроки:

Перехоплення HTTP-відповідей оригінальної програми і введення фальшивої функції JavaScript раніше ніж доставляється відповідь клієнту. Функція буде містити привабливу змінну з новостворені підроблені URL-адреси

Перехоплення HTTP-запитів та перевірка, чи запит орієнтований на фальшиву URL-адресу, яку було введено всередині змінної JavaScript у попередніх запитах. Якщо виявлено запит на фальшиву URL-адресу, то система повинна:

Запустити сигнал виявлення вторгнення, збільшити лічильник оповіщень і ввести дані запиту в файл журналу

Повернення відповіді залежно від налаштованого режиму:

Режим продовження: повертає дійсну HTTP-відповідь з 200 кодом стану, і щонайменше містить нову додаткову фальшиву URL-адресу, доступну атакувальнику.

Режим блокування: повернення дійсної відповіді HTTP, що містить спеціальну сторінку відповіді що система знаходиться на обслуговуванні.

2.2 Висновки до другого розділу

Запропонована у дипломній роботі методика враховує вразливості зі списку OWASP Top – 10. Від існуючих методик ця методика відрізняється тим, що вона є цілком новою і за найменших вмінь та навиків може бути впровадженна до будь якої системи захисту веб додатків. Завдяки її простоті ця методика не буде перешкоджати роботі веб додатків, а тому час простою буде мінімальним. Приклади наведені у розділі підтверджують доцільність її використання. У запропонованій методиці використовується чотири етапи по впровадженню її до системи захисту веб додатків.

РОЗДІЛ 3
ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ ВИКОРИСТАННЯ РОЗРОБЛЕНОЇ ОБМАННОЇ МЕТОДИКИ ДЛЯ ЗАХИСТУ ВЕБ-ДОДАТКІВ

3.1 Розрахунок економічної ефективності використання розробленої методики

Передбачається, що використання запропонованої методики дозволить зменшити витрати на при атаках на веб-додатки.

Запропонована методика дозволяє вводити обманну методику спеціалістам з базовим рівнем підготовки у галузі захисту інфомації.

Для розрахунку витрат на заробітну плату спеціаліста, що вводить обманну методику, нобхідно визначити:

– час на ознайомлення з технічним завданням та методикою, $t_{тз}$;

– час на підготовку середовища для виконання обманної методики (встановлення необхідних програм та їх налаштування), $t_{п}$;

– час на виконання етапів методики, $t_{м}$;

Перш за все необхідно обчислити умовну кількість етапів у методиці з урахуванням можливих уточнень у процесі роботи з методикою:

$$Q = q \cdot c\,(1 + p), \text{ штук}, \qquad (3.1)$$

де q – очікувана кількість етапів методики;

c – коефіцієнт складності методики, c = 1,25...2,0;

p – коефіцієнт корекції методики в процесі її опрацювання (p = 0,05...0,1, що відповідає внесенню 3...5 корекцій і переробці 5-10% методики в залежності від архітектури веб – додатка).

Кількість етапів введення обманної методики:

– методика, що запропонована у дипломній роботі – 4;

Оскільки розрахунок економічної ефективності проводиться з врахуванням можливості введення обманної системи спеціалістом з базовим рівнем підготовки

у галузі захисту інфомації та можливих корекцій або переробки методики в залежності від архітектури веб – додатка, необхідні коефіцієнти мають наступне значення:

– коефіцієнт складності методики, с = 2,0;

– коефіцієнт корекції методики в процесі її опрацювання р = 0,1; Умовна кількість етапів для методик має наступні значення для методики, що запропонована у дипломній роботі:

$$Q = 4 \cdot 2{,}0\ (1 + 0{,}1) = 8{,}8 \text{ шт.}$$

Час на опрацювання технічного завдання, додаткової літератури та ознайомлення з методикою можливо оцінити за формулою :

$$t_{\text{в}} = \frac{Q \cdot B}{75 \ldots 85 \cdot k}, \text{годин} \qquad (3.2)$$

де В – коефіцієнт збільшення тривалості етапу внаслідок недостатнього опису завдання, В = 1,2...1,5;

k – коефіцієнт, що враховує кваліфікацію програміста і визначається стажем роботи за фахом:

– до 2 років – 0,8;

– від 2 до 3 років – 1,0;

– від 3 до 5 років – 1,1...1,2;

– від 5 до 7 років – 1,3...1,4;

– понад 7 років – 1,5...1,6.

Таким чином, час на опрацювання технічного завдання, додаткової літератури та ознайомлення з методикою має наступні значення:

$$t_{\text{в}} = \frac{8{,}8 \cdot 1{,}5}{85 \cdot 0{,}8} = 0{,}19 \text{ години}$$

Час на підготовку середовища для введення обманної методики обчислюється за фоомулою:

$$t_{\text{а}} = \frac{Q}{20 \ldots 25 \cdot k}, \text{годин} \qquad (3.3)$$

Враховуючи це, обчислюємо час на підготовку середовища для виконання тесту на проникнення для методики, що запропонована у дипломній роботі:

$$t_{\text{а}} = \frac{8{,}8}{25 \cdot 0{,}8} = 0{,}44 \text{ години}$$

За наявною інформацією, тривалість введення обманної методики складає 1 – 2 тижні. 2 робочі тижні при 5 – денному робочому тижні становлять 10 днів або 80 годин при 8 – годинному робочому дні. Приймаючи до уваги особливості методики, що запропонована у дипломній роботі, орієнтовна тривалість тестування на проникнення сладає 40 годин. Отже, тривалість тестування за методикою має значення:

$$t_{\text{м}} = 40 \text{ годин}$$

Загальний час, що має бути витрачений на виконання тестування на проникнення за запропонованою методикою має значення для методики, що запропонована у дипломній роботі:

$$t_{\text{з}} = 40 + 0{,}44 + 0{,}19 = 40{,}63 \text{ годин}$$

Розмір заробітної плати спеціаліста з захисту інформації в Україні складає 15 тис. грн. на місяць. Враховуючи те, що середня кількість робочих годин за рік складає 166 годин, одна година робочого часу спеціаліста з захисту інформації

складає:

$$З_{зп} = \frac{15000}{166} = 90{,}4 \text{ грн/год}$$

де $З_{зп}$– заробітна плата спеціаліста з захисту інформації за одну годину

Таким чином, витрати на заробітну плату спеціаліста з захисту інформації за тестування конкретного веб – додатку, обчислюються за формулою:

$$К_з = З_{зп} \cdot t_з \qquad (3.4)$$

де $K_з$ – витрати на заробітну плату спеціаліста з захисту інформації, грн.

Витрати на заробітну плату спеціаліста з захисту інформації складають для методики, що запропонована у дипломній роботі:

$$К_з = З_{зп} \cdot t_з = 90{,}4 \cdot 40{,}63 = 3672{,}95 \text{ грн}$$

Таким чином, використання запропонованої методики дозволяє за мінімальних витрат знизити ризик викрадення критичної інформації або нанесення шкоди веб-додатку.

Результати розрахунків вказано у таблиці 3.1.

Таблиця 3.1

Таблиця, що характеризує витрати на проведення обманної методики

Кількість етапів тестування на проникнення у методиці, шт.	4
Час, витрачений на виконення тестування за методикою, годин	40,63

Витрати на заробітну плату спеціаліста з захисту інформації, грн.	3672,95

Враховуючи зменшений ризик вторгнення, зменшуються витрати на оплату роботи спеціаліста з захисту інформації за тестування одного конкретного веб – додатка [75].

3.2 Розрахунок витрат на створення програмного продукту

Витрати на створення програмного продукту **Кпз** складаються з витрат на заробітну плату виконавця програмного забезпечення **Ззп** і вартості витрат машинного часу, що необхідний для опрацювання програми на ПК **Змч**.

Зпр = 90,4 грн/год – средньогодинна заробітна плата програміста з нарахуваннями.

Вартість машинного часу для налагодження програми на ПК визначається за формулою:

$$З_{мч} = t_{опр} \cdot C_{мч} + t_{д}, \text{ грн,}$$

$$З_{мч} = 40{,}63 * 78{,}72 + 2{,}1 = 3200{,}50 \qquad (3.5)$$

де $t_{опр}$ – трудомісткість налагодження програми на ПК, годин;

$t_д$ – трудомісткість підготовки документації на ПК, годин;

$C_{мч}$ – вартість 1 години машинного часу ПК, грн./година.

Вартість 1 години машинного часу ПК визначається за формулою:

$$C_{мч} = P \cdot t_{нал} \cdot C_{е} + \frac{Ф_{зал} \cdot Н_{а}}{F_{р}} + \frac{К_{лпз} \cdot Н_{апз}}{F_{р}}, \text{ грн,}$$

$$C_{мч} = 0{,}2 * 3 * 1{,}2 + \frac{1000*100}{1920} + \frac{500*100}{1920} = 78{,}72 \qquad (3.6)$$

Де:

Р – встановлена потужність ПК, кВт;

$С_е$ – тариф на електричну енергію, грн/кВт·година;

$Ф_{зал}$ – залишкова вартість ПК на поточний рік, грн.;

$Н_а$ – річна норма амортизації на ПК, частки одиниці;

$Н_{апз}$ – річна норма амортизації на ліцензійне програмне забезпечення, частки одиниці;

$К_{лпз}$ – вартість ліцензійного програмного забезпечення, грн.;

$F_р$ – річний фонд робочого часу (за 40-годинного робочого тижня $F_р = 1920$.

Капітальні (фіксовані) витрати на проектування та впровадження проектного варіанта системи інформаційної безпеки складають:

$$К = К_{пр} + К_{зпз} + К_{пз} + К_{аз} + К_{навч} + К_н ,$$

$$К=1000+500+3673= 5172 \qquad (3.7)$$

Де:

– $К_{пр}$ – вартість розробки проекту інформаційної безпеки та залучення для цього зовнішніх консультантів, тис. грн;

– $К_{зпз}$ – вартість закупівель ліцензійного основного й додаткового програмного забезпечення (ПЗ), тис. грн;

– $К_{пз}$ – вартість створення основного й додаткового програмного забезпечення, тис. грн;

– $К_{аз}$ – вартість закупівлі апаратного забезпечення та допоміжних матеріалів, тис. грн;

– $К_{навч}$ – витрати на навчання технічних фахівців і обслуговуючого персоналу, тис. грн;

– $К_н$ – витрати на встановлення обладнання та налагодження системи інформаційної безпеки, тис. грн.

3.3 Оцінка можливого збиту від атаки (злому) на вузол або сегмент корпоративної мережі

Кінцевим результатом впровадження й проведення заходів щодо забезпечення інформаційної безпеки є величина відвернених втрат, що розраховується, виходячи з імовірності виникнення інциденту інформаційної безпеки й можливих економічних втрат від нього. По суті, ця величина відображає ту частину прибутку, що могла бути втрачена.

3.3.1 Оцінка величини збитку

Для розрахунку вартості такого збитку застосуємо наступну спрощену модель оцінки.

Необхідні вихідні дані для розрахунку:

$t_п$ – час простою вузла або сегмента корпоративної мережі внаслідок атаки, годин;

$t_в$ – час відновлення після атаки персоналом, що обслуговує корпоративну мережу, годин;

$t_{ви}$ – час повторного введення загубленої інформації співробітниками атакованого вузла або сегмента корпоративної мережі, годин;

$З_о$ – заробітна плата обслуговуючого персоналу (адміністраторів та ін.), грн на місяць;

$З_с$ – заробітна плата співробітників атакованого вузла або сегмента корпоративної мережі, грн на місяць;

$Ч_о$ – чисельність обслуговуючого персоналу (адміністраторів та ін.), осіб.;

$Ч_с$ – чисельність співробітників атакованого вузла або сегмента корпоративної мережі, осіб.;

О – обсяг продажів атакованого вузла або сегмента корпоративної мережі, грн у рік;

$П_{зч}$ – вартість заміни встаткування або запасних частин, грн;

I – число атакованих вузлів або сегментів корпоративної мережі;

N – середнє число атак на рік.

Упущена вигода від простою атакованого вузла або сегмента корпоративної мережі становить:

$$U = П_п + П_в + V,$$
$$U = 568{,}18+2420+27000 = 29988.18 \qquad (3.8)$$

Де: $П_п$ – оплачувані втрати робочого часу та простої співробітників атакованого вузла або сегмента корпоративної мережі, грн;

$П_в$ – вартість відновлення працездатності вузла або сегмента корпоративної мережі (переустановлення системи, зміна конфігурації та ін.), грн;

V – втрати від зниження обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі, грн.

Втрати від зниження продуктивності співробітників атакованого вузла або сегмента корпоративної мережі являють собою втрати їхньої заробітної плати (оплата непродуктивної праці) за час простою внаслідок атаки:

$$П_П = \frac{\sum Зс}{F} \cdot t_П \quad П_п = \frac{\sum Зс}{F} \cdot t_п,$$
$$П_п = \frac{25000}{176} * 4 = 568{,}18 \qquad (3.9)$$

де F – місячний фонд робочого часу (при 40-а годинному робочому тижні становить 176 ч).

Витрати на відновлення працездатності вузла або сегмента корпоративної мережі включають кілька складових:

$$П_в = П_{ви} + П_{пв} + П_{зч},$$
$$П_в = 1000 + 1420 = 2420 \qquad (3.10)$$

де $П_{ви}$ – витрати на повторне уведення інформації, грн.

$П_{пв}$ – витрати на відновлення вузла або сегмента корпоративної мережі.

$П_{зч}$ – вартість заміни устаткування або запасних частин, грн.

Витрати на повторне введення інформації $П_{ви}$ розраховуються виходячи з розміру заробітної плати співробітників атакованого вузла або сегмента корпоративної мережі $З_с$, які зайняті повторним введенням втраченої інформації, з урахуванням необхідного для цього часу $t_{ви}$:

$$П_{ВИ} = \frac{\sum Зс}{F} \cdot t_{ви}$$

$$П_{ви} = \frac{25000}{176} * 10 = 1420 \qquad (3.11)$$

Витрати на відновлення вузла або сегмента корпоративної мережі $П_{пв}$ визначаються часом відновлення після атаки $t_в$ і розміром середньо-годинної заробітної плати обслуговуючого персоналу (адміністраторів):

$$П_{ПВ} = \frac{\sum Зо}{F} \cdot t_в$$

$$П_{пв} = \frac{1800}{176} * 7 = 715 \qquad (3.12)$$

Втрати від зниження очікуваного обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі визначаються виходячи із середньогодинного обсягу продажів і сумарного часу простою атакованого вузла або сегмента корпоративної мережі:

$$V = \frac{О}{Fг} \cdot (t_П + t_в + t_{ви})$$

$$V = \frac{200000}{2080} * (4 + 7 + 10) = 27000 \qquad (3.13)$$

де $F_г$ – річний фонд часу роботи організації (52 робочих тижні, 5-ти денний робочий тиждень, 8-ми годинний робочий день) становить близько 2080 ч.

Таким чином, загальний збиток від атаки на вузол або сегмент корпоративної мережі організації складе:

$$B = \sum_i \sum_n U .$$

$$B = 27000 \qquad (3.14)$$

3.3.2 Загальний ефект від впровадження системи інформаційної безпеки

Загальний ефект від впровадження системи інформаційної безпеки визначається з урахуванням ризиків порушення інформаційної безпеки і становить:

$$E = B \cdot R - C,$$

$$E = 27000*0,4 -5000 = 5800 \qquad (3.15)$$

де B – загальний збиток від атаки на вузол або сегмент корпоративної мережі, тис. грн;

R – очікувана імовірність атаки на вузол або сегмент корпоративної мережі, частки одиниці;

C – щорічні витрати на експлуатацію системи інформаційної безпеки, тис. грн.

3.4 Визначення та аналіз показників економічної ефективності системи інформаційної безпеки

Оцінка економічної ефективності системи захисту інформації, розглянутої у спеціальній частині дипломного проекту, здійснюється на основі визначення та аналізу наступних показників:

– сукупна вартість володіння (TCO);

– коефіцієнт повернення інвестицій (ROI). У сфері інформаційної безпеки йому відповідає показник ROSI (Return on Investment for Security);

– термін окупності капітальних інвестицій T_o.

Коефіцієнт повернення інвестицій ROSI показує, скільки гривень додаткового прибутку приносить одна гривня капітальних інвестицій на впровадження системи інформаційної безпеки.

Щодо до інформаційної безпеки говорять не про прибуток, а про запобігання можливих втрат від атаки на сегмент або вузол корпоративної мережі, а отже:

$$ROSI = \frac{E}{K}, \quad \text{частки одиниці,}$$

$$\text{ROSI} = \frac{5800}{5172} = 1{,}12 \qquad (3.14)$$

Де: Е – загальний ефект від впровадження системи інформаційної безпеки (розділ 3.2 методичних вказівок, формула 3.8), тис. грн;

К – капітальні інвестиції за варіантами, що забезпечили цей ефект, тис. грн.

Для вибраного варіанта визначається розрахунковий строк окупності капітальних інвестицій $T_р$.

Термін окупності капітальних інвестицій $T_о$ показує, за скільки років капітальні інвестиції окупляться за рахунок загального ефекту від впровадження системи інформаційної безпеки:

$$T_o = \frac{K}{E} = \frac{1}{ROSI}, \quad \text{років}$$

$$T_o = \frac{37216}{14703} = \frac{1}{1{,}12} = 0{,}9 \text{ років} \qquad (3.17)$$

Якщо варіанти економічно рівноцінні, то приймається варіант, що забезпечує більш високу надійність, поліпшення умов праці.

3.4 Висновки до третього розділу

Було проаналізовано тривалість введення обманної методики. Затрати часу на введення обманної методики складає 40,63 години. Також було враховано витрати на оплату роботи спеціаліста з захисту інформації за тестування одного конкретного веб – додатка. Згідно розрахунків, витрати на заробітну плату і введення складають близько 5800 грн. за запропонованою у дипломній роботі методикою. Оскільки

взлам системи безпеки веб додатків може привести до критичних витрат у розмірі 29988.18 грн, тому витрати у розмірі 5800 грн повністю себе оправдують.

Введення запропонованої методики зможе окупити себе менш ніж за 1 рік після введення в експлуатацію.

Тому дані з розрахунків підтверджують економічну доцільність використання запропонованої у дипломній роботі методики для захисту веб – додатків.

ВИСНОВКИ

У дипломній роботі було досліджено можливість підвищення ефективності захисту веб додатків на прикладному рівні мережевої моделі. Для досягнення поставленої мети був виконаний аналіз сучасного стану захищеності веб – додатків. Оскільки пошук вразливостей є необхідним етапом, було визначено поширені вразливості веб – додатків, особливості атак через існуючі вразливості. Впровадження систем захисту баз даних мінімізує фінансові та репутаційні втрати, які можуть виникнути внаслідок крадіжки інформації. Система захисту баз даних запобігає несанкціонованому доступу до інформації в веб додатках, забезпечує контроль і фільтрацію типів взаємодій між користувачами та додатками, виявляє спроби несанкціонованого доступу до даних і перебиває їх, а також вирішує багато інших завданнь. За результатами досліджень було розроблено обманну методику захисту веб – додаткі, яка дозволяє зменшити вплив найбільш поширених вразливостей. З метою підтвердження економічної доцільності використання запропонованої у дипломній роботі методики, було розраховано вартість проведення тестування на проникнення за цією методикою. Враховуючи результати розрахунків, витрати на заробітну плату складають 3672,1 грн. за запропонованою у дипломній роботі методикою, а сама методика значно знижує ризик взламу системи і вірогідніть критичних витрат, що підтверджує економічну доцільність використання методики, що запропонована у дипломній роботі. Практичне значення роботи полягає у зниженні вірогідності зламу системи захисту веб додатку. Результати здійснених у дипломній роботі досліджень можуть бути використані при захисті веб – додатків та проведенні лабораторних робіт спеціалізованих курсів спеціальності «Кібербезпека».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Web application Wikipedia [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Web_application
2. Angela Stringfellow «What is Web Application Architecture» [Електронний ресурс]. – Режим доступу: https://stackify.com/web-application-architecture/
3. Стаття «Призначення та функції веб - сервера» [Електронний ресурс]. – Режим доступу: http://zametkinapolyah.ru/servera-i-protokoly/http-server-ili-veb- server-naznachenie-funkcii-i-rol-servera-v-http.html
4. Timothy Abel «Different types web application» [Електронний ресурс]. – Режим доступу: https://www.linkedin.com/pulse/6-different-types-web-application-development-timmothy-abel
5. Міжнародний стандарт ISO/IEC 29147 [Електронний ресурс]. – Режим доступу:http://standards.iso.org/ittf/PubliclyAvailableStandards/c045170_ISO_IEC_ 29147_2014.zip#en
6. Міжнародний стандарт ISO/IEC 27000 [Електронний ресурс]. – Режим доступу:http://standards.iso.org/ittf/PubliclyAvailableStandards/c066435_ISO_IEC_ 27000_2016(E).zip
7. Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу НД ТЗІ 1.1-003-99 [Електронний ресурс]. – Режим доступу:http://iszzi.kpi.ua/images/Info_bezpeka/ND_TZI/4_НД_ТЗІ_1.1-003-99.pdf
8. The Web Application Security Consortium [Електронний ресурс]. – Режим доступу: http://www.webappsec.org
9. Web Application Security Consortium – Класифікація загроз [Електронний ресурс]. – Режим доступу: http://projects.webappsec.org/f/WASC-TC-v2_0.pdf
10. Офіційний сайт проекту Common Weakness Enumeration [Електронний ресурс]. – Режим доступу: https://cwe.mitre.org/about/index.html
11. Документ Common Weakness Enumeration [Електронний ресурс]. – Режим доступу: https://cwe.mitre.org/data/published/cwe_v3.0.pdf
12. Офіційний сайт проекту Open Web Application Security Project [Електронний

ресурс]. – Режим доступу: https://www.owasp.org/index.php/ About_The_Open_Web_Application_Security_Project

13. Рейтинг вразливостей OWASP Top – 10 – 2017 Електронний ресурс]. – Режим доступу: https://www.owasp.org/images/7/72/OWASP_Top_10- 2017_%28 en%29.pdf.pdf

14. Veracode manual penetration testing [Електронний ресурс]. – Режим доступу: https://www.veracode.com/services/penetration-testing

15. Статья «Внедрение SQL кода» [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Внедрение_SQL-кода

16. Дэйв Уайтлегг, статья «Проверьте ваши приложения на уязвимости из списка OWASP Top 10 за 2013 год» [Електронний ресурс]. – Режим доступу: https://www.ibm.com/developerworks/ru/library/se-owasp-top10/index.html

17. Broken Authentication [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/Top_10-2017_A2-Broken_Authentication

18. Sensitive data exposure [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/Top_10-2017_A3-Sensitive_Data_Exposure

19. External Entity Attack [Електронний ресурс]. – Режим доступу: https://www.owasp.org/images/5/5d/XML_Exteral_Entity_Attack.pdf

20. Ian Muscat - What is XML External Entity [Електронний ресурс]. – Режим доступу:https://www.acunetix.com/blog/articles/xml-external-entity-xxe-vulnerabilit ies/

21.https://www.owasp.org/index.php/Top_10-2017_A5-Broken_Access_Control

22.https://www.owasp.org/index.php/Category:Access_Control

23.https://www.owasp.org/index.php/Top_10-2017_A6-Security_Misconfiguration

24. Cross – site scripting attack [Електронний ресурс]. – Режим доступу: https://www.acunetix.com/websitesecurity/cross-site-scripting/

25. Блог компанії Acunetix, стаття «What is insecure deserialization ?» [Електронний ресурс]. – Режим доступу: https://www.acunetix.com/blog/ articles/what-is-insecure-deserialization/

26. Стаття Insecure_Deserialization Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/Top_10-2017_A8-Insecure_Deserialization

27. Стаття Insufficient_Logging and Monitoring [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/Top_10-2017_A10Insufficient_Logging %26Monitoring
28. Офіційний сайт організації CERT [Електронний ресурс]. – Режим доступу: https://www.cert.org/about/
29. Офіційний сайт організації TF-CSIRT [Електронний ресурс]. – Режим доступу: https://tf-csirt.org/tf-csirt/
30. Офіційний сайт Державної служби спеціального зв'язку та захисту інформації: [Електронний ресурс]. – Режим доступу: http://www.dsszzi.gov.ua /dsszzi/control/uk/publish/article?showHidden=1&art_id=101338&cat_id=87062&cti me=1342168963970
31. Офіційний сайт організації CERT – UA [Електронний ресурс]. – Режим доступу: https://cert.gov.ua/
32. Офіційний сайт Національного банку України [Електронний ресурс]. – Режим доступу: https://bank.gov.ua/control/uk/publish/article?art_id=37602713
33. Офіційний сайт Дніпровської міської ради [Електронний ресурс]. – Режим доступу: https://dniprorada.gov.ua/uk/articles/item/22004/u-dnipri-vidkrili-centr-reaguvannya-na-kiberataki
34. Указ Президента України про рішення Ради національної безпеки і оборони України від 27 січня 2016 року "Про Стратегію кібербезпеки України" [Електронний ресурс]. – Режим доступу: http://zakon5.rada.gov.ua /laws/show/96 /2016
35. Офіційний сайт форуму команд реагування на компютерні надзвичайні події [Електронний ресурс]. – Режим доступу: https://www.first.org/about/
36. Офіційний сайт проекту Common Vulnerability Scoring System [Електронний ресурс]. – Режим доступу: https://www.first.org/cvss/
37. Common Vulnerability Scoring System v3.0: Specification Document [Електронний ресурс]. – Режим доступу: https://www.first.org/cvss/specification-document

38.https://cve.mitre.org/about/index.html
39.https://cve.mitre.org/about/faqs.html#what_is_cve_id
40.https://cve.mitre.org/about/faqs.html#MITRE_role_in_cve
41.https://cve.mitre.org/cve/cna.html
42. CVE Numbering authorities [Електронний ресурс]. – Режим доступу: https://cve.mitre.org/cve/request_id.html
43. База даних вразливостей CVE Details [Електронний ресурс]. – Режим доступу: https://www.cvedetails.com
44. База даних вразливостей NVD [Електронний ресурс]. – Режим доступу: https://nvd.nist.gov/vuln
45. База даних вразливостей Vulnerability Notes Database [Електронний ресурс]. – Режим доступу: http://www.kb.cert.org/vuls
46. База даних вразливостей SecurityFocus [Електронний ресурс]. – Режим доступу: https://www.securityfocus.com/
47. Міжнародний стандарт ISO/IEC 29147 [Електронний ресурс]. – Режим доступу:http://standards.iso.org/ittf/PubliclyAvailableStandards/c045170_ISO_IEC_29147_2014.zip#en
48. Документ Symantec Software Security Vulnerability Management Process [Електронний ресурс]. – Режим доступу: https://www.symantec.com/content/en/us/global/security_response/data/security/Symantec_Software_Security_Vulnerability_Management_Process.pdf
49. Bug Bounty List [Електронний ресурс]. – Режим доступу: https://www.bugcrowd.com/bug-bounty-list/
50. А.В.Лукацький - «Як працює сканер безпеки» [Електронний ресурс]. – Режим доступу: http://citforum.ck.ua/internet/securities/scaner.shtml
51. Довідкове керівництво сканера nmap [Електронний ресурс]. – Режим доступу: https://nmap.org/man/ru/index.html
52.https://portswigger.net/burp/help/suite_gettingstarted

53.https://metasploit.help.rapid7.com/docs/metasploit-basics
54. Офіційний сайт виробника Kali Linux [Електронний ресурс]. – Режим доступу: https://www.kali.org
55. Офіційний сайт виробника BlackArch [Електронний ресурс]. – Режим доступу: https://blackarch.org/index.html
56. Офіційний сайт виробника ParrotSecurity [Електронний ресурс]. – Режим доступу: https://www.parrotsec.org
57. Офіційний сайт виробника BlackBox [Електронний ресурс]. – Режим доступу:https://backbox.org
58. Офіційний сайт виробника сканера Nessus [Електронний ресурс]. – Режим доступу: https://www.tenable.com/products/nessus/nessus-professional
59. Офіційний сайт компанії Агенство активного аудита [Електронний ресурс]. – Режим доступу: http://auditagency.com.ua/?r=blog&p=Pentest&lang=ru
60. Методика тестування OWASP Testing guide [Електронний ресурс]. – Режим доступу: https://www.owasp.org/images/1/19/OTGv4.pdf
61. Методика тестування PTES [Електронний ресурс]. – Режим доступу: http://www.pentest-standard.org/index.php/Main_Page
62. Information Systems Security Assessment Framework [Електронний ресурс]. – Режим доступу: http://www.oissg.org/files/issaf0.2.1.pdf
63. База даних експлойтів веб – додатків [Електронний ресурс]. – Режим доступу: https://www.exploit-db.com/webapps/
64. Стаття «Metasploit инструкция по применению» [Електронний ресурс]. – Режим доступу: https://cryptoworld.su/metasploit-%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F-%D0%BF%D0%BE-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8E/
65. Using Burp Proxy [Електронний ресурс]. – Режим доступу: https://support.portswigger.net/customer/portal/articles/1783119-using-burp-proxy

66. Сайт виробника сканера sqlmap [Електронний ресурс]. – Режим доступу: http://sqlmap.org
67. Офіційний сайт виробника сканера Burp Suite, тестування XSS – атак [Електронний ресурс]. – Режим доступу: https://support.portswigger.net/customer/portal/articles/2325939-Methodology_Attacking%20Users_XSS_Using%20Burp%20to%20Manually%20Test%20For%20Reflected%20XSS.html
68. Testing for Weak SSL/TLS Ciphers [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/Testing_for_Weak_SSL/TLS_Ciphers,_Insufficient_Transport_Layer_Protection_(OTG-CRYPST-001)
69. Офіційний сайт виробника сканера nmap [Електронний ресурс]. – Режим доступу:https://nmap.org/nsedoc/scripts/ssl-enum-ciphers.html
70. Офіційний сайт компаній Berzha Security [Електронний ресурс]. – Режим доступу: https://berezhasecurity.com/index_uk.html
71. О.Г. Вагонова, Ю.О. Волотковська, Н.М. Романюк. – Методичні вказівки до виконання економічної частини дипломного проекту для студентів напряму підготовки 1701 Інформаційна безпека
72. Penetration testing, frequently asked questions [Електронний ресурс]. – Режим доступу: https://highbitsecurity.com/FAQ-penetrationtesting.php
73. Кодекс законів про праці України [Електронний ресурс]. – Режим доступу: http://zakon3.rada.gov.ua/laws/show/322-08/page3
74. Інформаія про заробітну плату спеціаліста з інформаційної безпеки [Електронний ресурс]. – Режим доступу: https://rabota.ua/zapros/%D1%81%D0%BF%D0%B5%D1%86%D0%B8%D0%B0
75. Норми робочого часу [Електронний ресурс]. – Режим доступу: http://vizitc.kiev.ua/ru/infokonsultant/normi_rabochego_vremeni.html

ДОДАТОК А. Відомість матеріалів дипломного проекту

№	Формат	Найменування	Кількість листів	Примітка
1	А4	Реферат	3	
2	А4	Список умовних скорочень	2	
3	А4	Зміст	2	
4	А4	Вступ	1	
5	А4	1 Розділ	24	
6	А4	2 Розділ	6	
7	А4	3 Розділ		
8	А4	Висновки		
9	А4	Перелік посилань		
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	

ДОДАТОК Б Копії наукових публікацій

УДК 004.773.3

Ушенко М.С. студент гр. 125М-17-2
Науковий керівник: Галушко С.О.
(*Національний технічний університет "Дніпровська політехніка", м. Дніпро, Україна*)

ЗАХИСТ КОРПОРАТИВНОЇ ЕЛЕКТРОННОЇ ПОШТИ

Із зростанням ролі електронної пошти збільшуються і потреби в управлінні нею - це завдання в даний час актуальне, як ніколи. Електронна пошта для більшості компаній є основним засобом комунікації. Переписка необхідна як для спілкування всередині самої компанії, так і для обміну даними з партнерами, клієнтами, постачальниками, державними органами і т.д. Тим більшого значення набуває ефективна корпоративна система управління електронною поштою - надійна, з високою пропускною здатністю, оснащена засобами контролю і відсіву небажаних повідомлень. Адміністраторам поштових систем доводиться постійно боротися зі спамом, вірусами та хакерськими атаками.

Електронна пошта має численні перевагами, але саме через ці достоїнства виникають основні ризики, пов'язані з її використанням. Наприклад, доступність електронної пошти перетворюється в недолік, коли користувачі починають застосовувати пошту для розсилки спаму, легкість у використанні та безконтрольність призводить до витоку інформації, можливість пересилання різних форматів документів - до поширення вірусів і т.д.

З огляду на описані вище ризики, пов'язані з використанням електронної пошти, організаціям необхідно вжити відповідних заходів для захисту від них. Підхід до захисту повинен бути всебічним і комплексним - необхідно поєднувати організаційні заходи з використанням відповідних технічних засобів. До організаційних заходів належать розробка та впровадження в компанії політики використання електронної пошти. Технічні засоби повинні забезпечити виконання даної політики як за рахунок моніторингу поштового трафіку, так і за рахунок адекватного реагування на порушення.

Відомі численні приватні рішення для аналізу трафіку або його суцільного сканування, фільтрації контента, блокування спаму, генерування попереджень про небезпеку, підтримки певних політик, балансу завантаження поштових серверів і т. д. Однак рішень, які об'єднують всі необхідні інструменти і працюють на різних платформах з різними поштовими системами, не так багато. Вважається що якісному антиспамовому продукту повинні бути притаманні такі основні властивості:

– інтероперабельність (здатність до взаємодії) - система управління легко вбудовується в існуючу інфраструктуру підприємства, здатна працювати в гетерогенному середовищі і має можливості масштабування;

– мінімальний вплив - система не повинна додавати ніяких нових ризиків, явних чи прихованих вимог додаткових ресурсів;
– засоби спостереження - вся інформація про стан системи і про те, як вона використовується, повинна бути під рукою у адміністратора;
– захист - система повинна вміти виявляти загрози в реальному часі і самостійно ліквідувати їх без втручання адміністратора; це стосується і вірусної загрози та спроб хакерських атак;
– надійність - мета роботи системи управління - підвищити ступінь готовності ресурсу і пом'якшити вплив відмов, певним чином обробляючи аварійні ситуації;
– прийнятна окупність - щоб отримати схвалення осіб, що дають згоду на інвестиції, система повинна мати мінімально можливу вартість.

Прикладом комплексного додаткового рішення може служити ПО, розроблене компанією Postini. Продукт розташовується між каналом Internet і внутрішньою мережею організації. він
не замінює собою основний поштовий сервер і не вносить жодних змін в сховище повідомлень.

Додаток обробляє SMTP-трафік на рівні сесії і не вимагає будь-якої інтеграції з самим поштовим сервером. При цьому виявляються поштові атаки, спам і віруси. Виконується також збір детальної статистичної інформації, що дає змогу аналізувати поштовий трафік компанії.
Компанії, які спеціалізуються в області боротьби зі спамом, добре знають більшість прийомів і вивертів спамерів. Тому в середньому рівень очищення потоку пошти у них вище, ніж можуть досягти адміністратори звичайних компаній, а відсоток відсіву потрібних ділових листів - менше. Вони швидко виявляють нові прийоми спамерів і проводять нейтралізуючі заходи. Така послуга називається спам-аутсорсинг.

Найбільш ефективним заходом перешкоди витоків інформації електронною поштою є її шифрування. В цьому випадку, навіть якщо зловмисник отримав доступ до пошти, прочитати її він не зможе.

При цьому методі використовуються два ключа - публічний і приватний, за допомогою яких листи шифруються і дешифруються. Одержувач повинен знати публічний ключ, для цього відправник може опублікувати Private key онлайн, наприклад на своєму сайті, або розіслати звичайним листом всім потенційним одержувачам.

ВИСНОВКИ

В даний час діяльність компаній все більше залежить від електронної пошти. Зручність і практичність електронної пошти очевидні. Однак не можна не враховувати проблеми, які виникають у зв'язку з її неконтрольованим використанням. Наслідки для компаній можуть бути непередбачуваними.

Таким чином, всі перераховані вище факти ще раз підтверджують необхідність застосування в системах безпеки корпоративних мереж систем контролю вмісту електронної пошти, які здатні не тільки забезпечити захист системи електронної пошти і стати ефективним елементом управління поштовим

потоком, але і значно підвищити ефективність діяльності підприємства чи організації.

Перелік посилань

1.Корпоративная электронная почта: контроль, защита, надежность [Електронний ресурс] Режим доступу: **https://itc.ua/articles/korporativnaya_jelektronnaya_pochta_kontrol_zashhita_nadezhnost_10992/**

2.Безопасность систем электронной почты [Електронний ресурс] Режим доступу: http://citforum.ck.ua/security/internet/email/article1.6.200355.html

3.Защита корпоративной почты [Електронний ресурс] Режим доступу: https://efsol.ru/articles/protection-of-corporate-e-mail.html

4.Безопасность электронной почты: шифрование писем [Електронний ресурс] Режим доступу: https://habr.com/post/190130/

ДОДАТОК В. Відгук керівника економічного розділу

Керівник : _____ к.е.н.,доц. Пілова Д.П.

(підпис)

ДОДАТОК Г. Відгук

на дипломний проект магістра

студента групи 125М-17-2

Ушенко Микити Сергійовича

з теми: «Аналіз захищеності веб-додатків від атак на прикладному рівні мережевої моделі OSI»

Метою дипломного проекту є забезпечення кібербезпеки веб додатків.

Тема дипломного проекту безпосередньо пов’язана з об’єктом діяльності магістра фаху 6.125 «Кібербезпека». Для досягнення поставленої мети в дипломному проекті вирішуються наступні задачі: визначити, які існують загрози кібербезпеки у веб додатках. Розглянути, які існують рішення кібербезпеки у веб додатках. Зібрати дані про угрози, проаналізувати існуючі рішення проблем. Виробити методику по забезпеченню безпеки веб додатків. Практичне значення результатів дипломного проекту полягає в можливості їх використання у будь яких веб додатках.

Перевагою дипломного проекту є розробка інструкцій які дозволяють встановити більш надійне програмне забезпечення.

За час дипломування Ушенко М.С. проявив себе фахівцем, здатним самостійно вирішувати поставлені задачі.

Оцінка роботи_____

Керівник дипломної роботи,
д.т.н., проф. Корнієнко В.І.

Керівник спец. розділу,

ст. викл. кафедри БІТ Галушко С.О.

ДОДАТОК Д. Перелік файлів на електронному носії

1. Дипломний проект Ушенко М.С. 125М-17-2 – Пояснювальна записка.
2. Ушенко М.С.pptx – Презентація.