ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Технолологія забезпечення безпеки Web-сервісів організації»

зі спеціальності *125 Кібербезпека та захист інформації*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Головань ВЛАДИСЛАВ
(підпис)

Виконав: здобувач вищої освіти групи БСДМ-62
ГОЛОВАНЬ Владислав
(прізвище, ім'я)

Керівник Кожухівський А.Д., д.т.н., професор
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

Кафедра Систем та технологій кібербезпеки
Ступінь вищої освіти Магістр
Спеціальність 125 Кібербезпека та захист інформації
Освітньо-професійна програма Інформаційна та кібернетична безпека

ЗАТВЕРДЖУЮ
Завідувач кафедри
Систем та технологій
кібербезпеки
_____ Галина ГАЙДУР
"___" грудня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

ГОЛОВАНЮ Владиславу Вікторовичу
(прізвище, ім'я)

1. Тема кваліфікаційної роботи: «**Технологія забезпечення безпеки Web-сервісів організації**»

керівник кваліфікаційної роботи Кожухівський А.Д., д.т.н., професор
(прізвище, ім'я, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024 року № 320.

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 15.12.2024 р.

3. Вихідні дані до кваліфікаційної роботи
інформаційні ресурси організації;
безпекове рішення для організації;
наукова та технічна література, експлуатаційна документація, нормативні документи, міжнародні стандарти.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження проблеми забезпечення безпечної роботи Web-сервісів організації
2. Аналіз методів та засобів забезпечення безпечної роботи Web-сервісів

організації

3. Технологія забезпечення безпечної роботи Web-сервісів організації

5. Перелік графічного матеріалу
Презентація PowerPoint.

6. Дата видачі завдання 01.10.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Визначення актуальності проблеми забезпечення безпечної роботи Web-сервісів організації.	01.10.2024 р.	
2.	Аналіз наукової та технічної літератури з питань теми кваліфікаційної роботи.	12.10.2024 р.	
3.	Аналіз методів та засобів забезпечення безпечної роботи Web-сервісів організації.	27.10.2024 р.	
4.	Технологія забезпечення безпечної роботи Web-сервісів організації.	03.11.2024 р.	
5.	Розроблення рекомендацій щодо застосування технології забезпечення безпечної роботи Web-сервісів організації.	15.11.2024 р.	
6.	Оформлення результатів дослідження.	26.11.2024 р.	
7.	Підготовка доповіді до захисту.	15.12.2024 р.	

Здобувач вищої освіти _____ Владислав Головань
(підпис) *(ім'я, прізвище)*

Керівник кваліфікаційної роботи _____ Кожухівський А.Д.
(підпис) *(ім'я, прізвище)*

РЕФЕРАТ

Текстова частина бакалаврської роботи: 62 сторінок, 2 рисунка, 11 джерел.

Об'єкт дослідження – процес виявлення та протидії несанкціонованому втручанню до Web-сервісів організації.

Предмет дослідження – методи і засоби протидії несанкціонованого доступу до Web-сервісів організації.

Мета роботи – розробити рекомендації щодо впровадження безпекових рішень для забезбечення захищеності Web-сервісів організації.

Методи дослідження – теорія інформації, стандарти в сфері кібербезпеки, безпека Web-сервісів.

В роботі приведено основні відомості про Web-сервіси організації. Проаналізовано різні види загроз, детально розглянуто засоби боротьби з ними та досліджено засоби їх протидії.

Досліджено методи, засоби боротьби з вразливостями Web-сервісів організації.

На основі досліджень проведених в роботі розроблено рекомендації щодо застосування методів та засобів інтеграції безпекових рішень. *Галузь використання* – кібербезпека.

КІБЕРБЕЗПЕКА, АУТЕНТИФІКАЦІЯ, ВРАЗЛИВОСТІ, ВЕБ-САЙТ, ВЕБДОДАТОК, АВТОРИЗАЦІЯ, МЕТОДИКА, ЗАХИСТ, ПОШИРЕНИЙ, СКАНЕР ВРАЗЛИВОСТЕЙ, DNS, НЕСАНКЦІОНОВАНЕ ВТРУЧАННЯ, ВІДМОВА В ОБСЛУГОВУВАННІ, КІБЕРАТАКА.

ABSTRACT

Bachelor's thesis: 62 pages, 2 draw, 11 sources

Object of research– the process of detecting and countering unauthorized interference with the organization's Web services.

Subject of research– methods and means of combating unauthorized access to the organization's Web services.

The aim of research – develop recommendations for the implementation of security solutions to ensure security for the organization's Web services.

Research methods - information theory, standards in the field of cyber security, network security.

The work contains basic information about the organization's Web services. Various types of threats were analyzed, the means of combating them were examined in detail, and the means of their countermeasures were studied.

The methods and means of combating the vulnerabilities of the organization's Web services were studied.

Based on the research carried out in the work, recommendations were developed for the use of methods and means of integrating security solutions.

Field of use–cybersecurity.

CYBERSECURITY, AUTHENTICATION, VULNERABILITIES, WEBSITE, WEB APPLICATION, AUTHORIZATION, METHODOLOGY, PROTECTION, ADVANCED, VULNERABILITY SCANNER, DNS, UNAUTHORIZED INTERFERENCE, DENIAL OF SERVICE, CYBERATTACK.

ЗМІСТ

Стор.

ВСТУП

Актуальність дослідження. Інформаційна безпека – важлива функція в будь-якій веб-програмі. Оскільки майже всі веб-програми відкриті для Інтернету, завжди існує ймовірність загрози безпеці веб-додатків. Отже, при розробці веб-додатків завжди рекомендується переконатися, що програма розроблена з урахуванням вимог безпеки. Щоб зрозуміти які загрози можуть бути, розглянемо простий сценарій вебзастосунку і розберемось, як він працює з точки зору безпеки.

Об'єкт дослідження– Web-сервіси організації.

Предмет дослідження – методи захисту Web-сервісів організації.

Мета роботи – розроблення рекомендацій щодо захисту Web-сервісів організації.

Завдання бакалаврської роботи:

- Проаналізувати існуючі вразливості Web-сервісів організації;
- Визначити наявні уязвимості Web-сервісів організації;
- дослідити існуючі засоби боротьби з небезпеками Web-сервісів організації;
- розробити рекомендації щодо забезпечення безпеки у Web-сервісах організації.

Методи дослідження – опрацювання технічної літератури за даною темою, аналіз експлуатаційної документації інструментів, порівняльний аналіз сервісів.

Практичне значення одержаних результатів: рекомендації щодо захисту Web-сервісів організації.

1 АНАЛІЗ ВРАЗЛИВОСТЕЙ WEB-СЕРВІСІВ

1.1 Поняття Web-сервісу

Вебслужба, вебсервіс [1].— програмна система, що ідентифікується URI, і публічні інтерфейси та прив'язки якої визначені та описані мовою XML. Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису з використанням повідомлень, що базуються на XML та передаються за допомогою інтернет-протоколів.

Web-сервіси не являються чимось новим і зазвичай приймають форму інтерфейсів прикладного програмування. У наш час конкуренція в бізнесі, в навчанні, в промисловості, в обміні інформацією – це щоденна потреба.

Термін «вебслужба» введено організацією W3C і застосовується до багатьох різних систем, але в основному термін стосується клієнтів та серверів, що взаємодіють за допомогою повідомлень протоколу SOAP. В обох випадках припускається, що існує також опис доступних операцій у форматі WSDL. Хоча наявність цього опису не є вимогою SOAP, а радше передумовою для автоматичного генерування коду на платформах Java та .NET на стороні клієнта.

На рисунку 1.1 зображена структура web-сервісу.

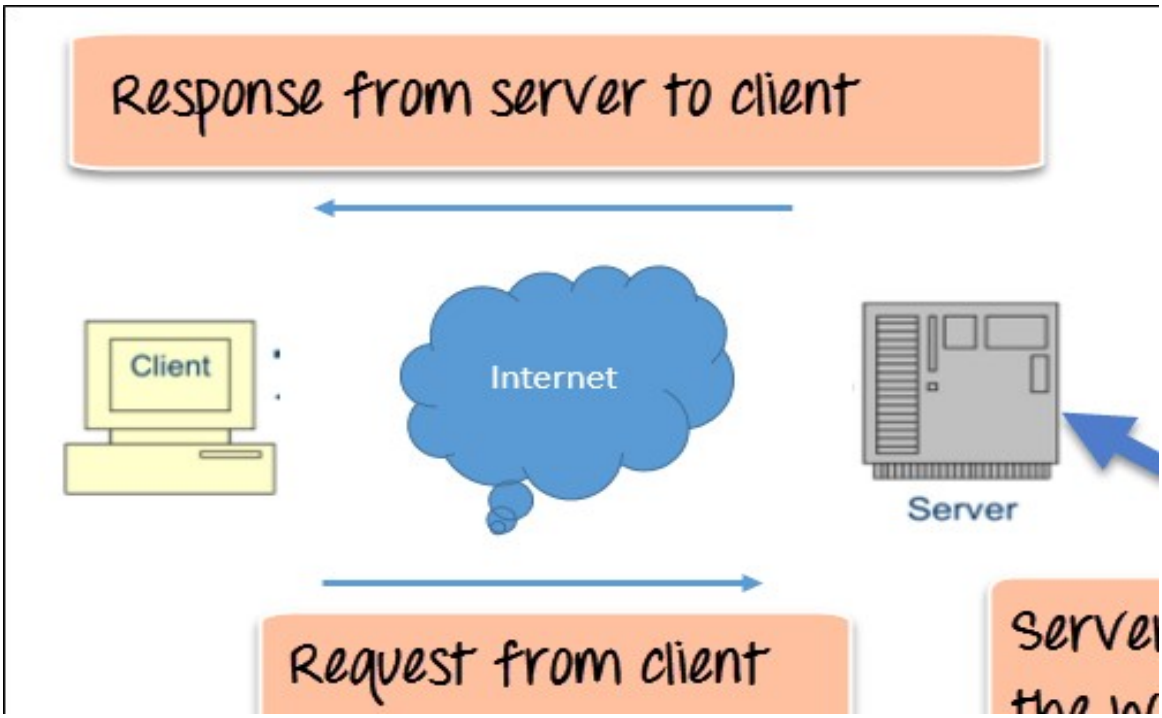

Рис. 1.1. Структура web-сервісу

Наведена вище діаграма показує дуже спрощену картину того, як насправді буде працювати веб-служба. Клієнт ініціював би серію викликів веб-сервісу через запити до сервера, на якому розміщувався б фактичний веб-сервіс.

Ці запити надсилаються за допомогою так званих віддалених викликів процедур. Виклики віддалених процедур (RPC) — це виклики методів, розміщених у відповідній веб-службі.

Як приклад, Amazon надає веб-сервіс, який надає ціни на продукти, що продаються онлайн через amazon.com. Інтерфейсний або презентаційний рівень може бути в .Net або Java але будь-яка мова програмування матиме можливість спілкуватися з веб-службою.

Основним компонентом дизайну веб-сервісу є дані, які передаються між клієнтом і сервером, тобто XML. XML (розширювана мова розмітки) є аналогом HTML і легкою для розуміння проміжною мовою, яку розуміють багато мов програмування.

Отже, коли програми спілкуються одна з одною, вони насправді спілкуються в XML. Це забезпечує спільну платформу для спілкування програм, розроблених на різних мовах програмування.

Веб-служби використовують протокол SOAP (Simple Object Access Protocol) для надсилання XML-даних між програмами. Дані надсилаються через звичайний HTTP. Дані, які надсилаються з веб-служби до програми, називаються повідомленням SOAP. Повідомлення SOAP — це не що інше, як документ XML. Оскільки документ написаний у форматі XML, клієнтська програма, що викликає веб-службу, може бути написана будь-якою мовою програмування.

Існує в основному два типи веб-сервісів:

1. Веб-сервіси SOAP.
2. Веб-сервіси RESTful.

Для повноцінної роботи веб-сервісу необхідно встановити певні компоненти. Ці компоненти мають бути присутніми незалежно від того, яка мова розробки використовується для програмування веб-сервісу.

SOAP (Простий протокол доступу до об’єктів) [2]. відомий як незалежний від транспорту протокол обміну повідомленнями. SOAP базується на передачі XML-даних у вигляді повідомлень SOAP. Кожне повідомлення має те, що називається документом XML. Лише структура XML-документа відповідає певному шаблону, але не вміст. Найкраща частина веб-служб і SOAP полягає в тому, що вони надсилаються через HTTP, який є стандартним веб-протоколом.

Ось з чого складається повідомлення SOAP

1. Кожен документ SOAP повинен мати кореневий елемент, відомий як елемент. Кореневий елемент є першим елементом у документі XML.
2. «Конверт» в свою чергу ділиться на 2 частини. Перший – це заголовок, а наступний – тіло.

3. Заголовок містить дані маршрутизації, які в основному є інформацією, яка повідомляє XML-документу, якому клієнту його потрібно надіслати.
4. Тіло міститиме фактичне повідомлення.

На рисунку 1.2 показаний простий приклад зв’язку через SOAP.

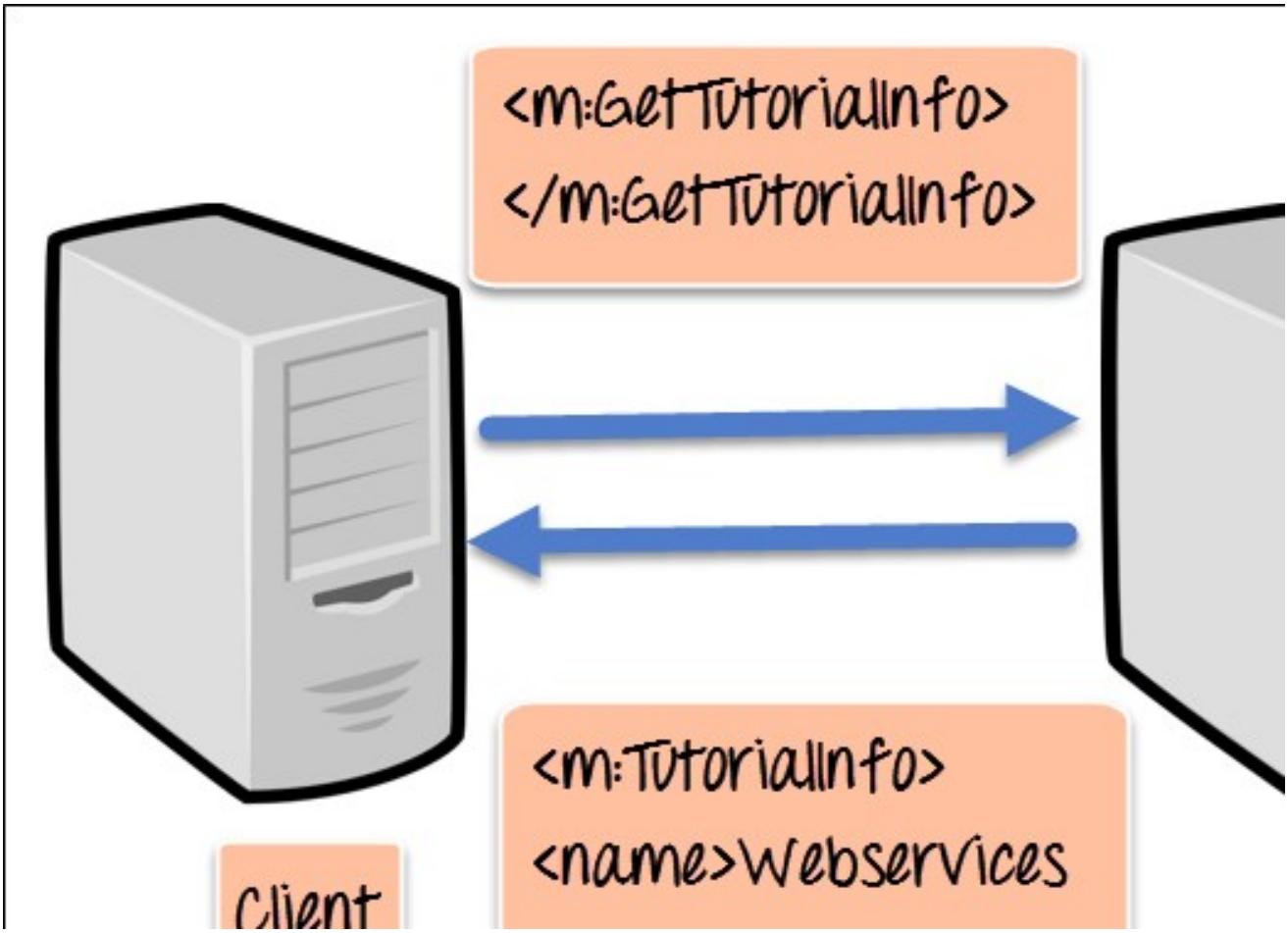

Рис. 1.2. Протокол SOAP

WSDL (мова опису веб-служб) [3].

Веб-службою не можна користуватися, якщо її неможливо знайти. Клієнт, який викликає веб-службу, повинен знати, де веб-служба фактично знаходиться.

По-друге, клієнтська програма повинна знати, що насправді робить веб-служба, щоб вона могла викликати правильну веб-службу. Це робиться за допомогою WSDL, відомого як мова опису веб-сервісів. Файл WSDL знову є файлом на основі XML, який в основному повідомляє клієнтській програмі, що

робить веб-служба. Використовуючи документ WSDL, клієнтська програма зможе зрозуміти, де знаходиться веб-служба та як її можна використовувати.

Universal Description, відкриття та інтеграція (UDDI)

UDDI — це стандарт для опису, публікації та пошуку веб-служб, які надаються певним постачальником послуг. Він надає специфікацію, яка допомагає розміщувати інформацію у веб-службах.

Web-сервіси Architектура

Кожен фреймворк потребує певної архітектури, щоб переконатися, що весь фреймворк працює належним чином, подібно до веб-служб. The **Web-сервіси Architектура** складається з трьох окремих ролей, як зазначено нижче:

1. **Provider** – Провайдер створює веб-сервіс і робить його доступним для клієнтської програми, яка хоче ним скористатися.
2. **Запитувач** – Запитувач – це не що інше, як клієнтська програма, якій потрібно зв’язатися з веб-службою. Клієнтською програмою може бути .Net, Javaабо будь-яка інша мовна програма, яка шукає якусь функціональність через веб-службу.
3. **брокер** – Брокер – це не що інше, як програма, яка надає доступ до UDDI. UDDI, як обговорювалося в попередній темі, дозволяє клієнтській програмі знаходити веб-службу.

На діаграмі нижче показано, як постачальник послуг, запитувач послуг і реєстр послуг взаємодіють один з одним.

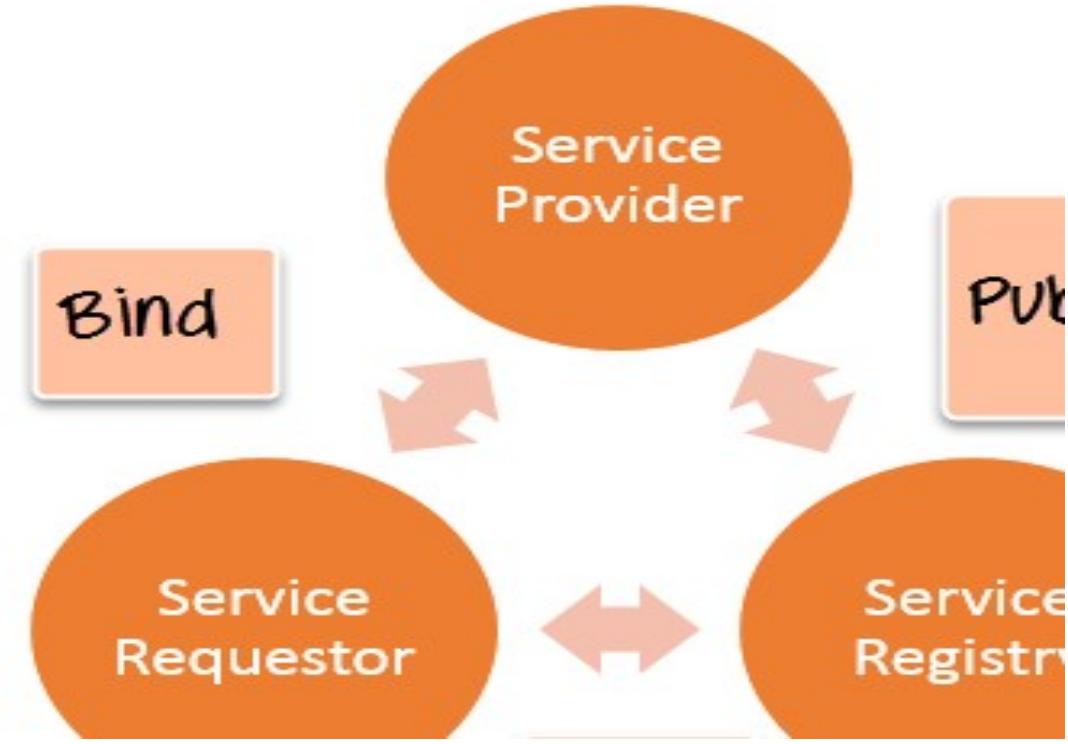

Рис. 1.3. Web-сервіси архітектура

1. **Публікувати** – Провайдер інформує брокера (реєстр послуг) про існування веб-сервісу, використовуючи інтерфейс публікації брокера, щоб зробити сервіс доступним для клієнтів
2. **Знайти** – Запитувач консультується з брокером, щоб знайти опубліковану веб-службу
3. **Bind** – За допомогою інформації, отриманої від брокера (реєстру послуг) про веб-службу, запитувач може прив'язати або викликати веб-службу.

Характеристики веб-сервісу

Веб-сервіси мають такі особливості поведінки:

1. Вони засновані на XML [4]. – Веб-служби використовують XML для представлення даних на рівнях представлення та транспортування даних. Використання XML усуває будь-яку залежність від мережі, операційної системи чи платформи, оскільки XML є спільною мовою, зрозумілою всім.

2. Слабко зчеплені – Слабко пов’язаний означає, що клієнт і веб-служба не пов’язані один з одним, що означає, що навіть якщо веб-служба змінюється з часом, це не повинно змінювати спосіб виклику клієнтом веб-служби. Прийняття слабозв’язаної архітектури робить програмні системи більш керованими та дозволяє простішу інтеграцію між різними системами.
3. Synchronous або Asynchronous функціональність - Syncхронічність відноситься до прив'язки клієнта до виконання послуги. У синхронних операціях клієнт фактично чекатиме, поки веб-служба завершить операцію. Прикладом цього, ймовірно, є сценарій, у якому виконується операція читання та запису бази даних. Якщо дані зчитуються з однієї бази даних, а потім записуються в іншу, то операції повинні виконуватися послідовно. Асинхронні операції дозволяють клієнту викликати службу, а потім паралельно виконувати інші функції. Це один із поширених і, мабуть, найбільш бажаних методів для забезпечення того, щоб інші служби не зупинялися під час виконання певної операції.
4. Можливість підтримки віддалених викликів процедур (RPC) – Веб-сервіси дозволяють клієнтам викликати процедури, функції та методи на віддалених об’єктах за допомогою протоколу на основі XML. Віддалені процедури надають вхідні та вихідні параметри, які має підтримувати веб-служба.
5. Підтримує обмін документами – Однією з ключових переваг XML є його загальний спосіб представлення не лише даних, але й складних документів. Ці документи можуть бути такими ж простими, як поточна адреса, або такими ж складними, як ціла книга.

1.2 Існуючі загрози для функціонування Web-сервісів

Загрози кібербезпеці веб-серверів викликають значне занепокоєння як у бізнесу, так і у приватних осіб. Скомпрометований веб-сервер може призвести до витоку даних, несанкціонованого доступу, пошкодження веб-сайту і навіть фінансових втрат.

З ростом кількості та значення різних веб-сервісів в суспільному житті, наприклад, таких як банківські послуги, покупка товарів он-лайн та ін., також зростає значення забезпечення їх безпеки. Ефективність забезпечення безпеки може зростати якщо знати звідки чекати загрози.

Найпростіший і поширений спосіб, яким користуються зловмисники - пошук і використання неправильної конфігурації сервера. Ці неправильні конфігурації часто відбуваються під час початкової настройки і можуть не виявлятися, поки зловмисники не спробують проникнути в корпоративні мережі. Боротьба з цією проблемою залежить від ретельного вибору постачальника - запросіть детальну інформацію про пропозиції безпеки і потенційні прогалини, що існують між хостинговими і локальними рішеннями, а також моніторинг програмного забезпечення для безпечного підключення до сервера.

У той час як зловмисники представляють критичні загрози безпеки сервера, більш вірогідні внутрішні загрози - особливо у випадках, коли співробітники випадково обмінюються конфіденційними даними або використовують небезпечні додатки. Це може відбуватися через недбальство, незнання або випадковості. Але нерідкі випадки інсайдерських погроз - це зловмисні спроби незадоволених або колишніх співробітників продати корпоративні дані. Проте, незалежно від причини, рішення залишається тим же: впроваджувати рішення для ведення журналів і моніторингу в масштабі всієї мережі, які повідомляють адміністраторів про ризиковану поведінку, у тому числі не варто забувати про такий процес як несанкціоновані запити на випуск SSL сертифікатів.

Створення неконтрольованих ресурсів підприємства з використання Wildcard сертифікатів SSL, залишаються поширеними серверними ризиками, особливо з урахуванням збільшення кількості небезпечних IoT-устройств, які можуть бути зламані хакерами і використані з недоброю метою.

Для багатьох організацій звичайною практикою є жорстке кодування облікових даних, таких як приватні ключі сертифікатів і дозволу доступу до мережі, у вихідний код або документацію. Якщо зловмисники розкривають ці дані, вони можуть використовувати їх для отримання привілейованого доступу до

мережі або продати їх для отримання прибутку в темній мережі. Зупиніть витоки облікових даних, зберігаючи їх у захищених сховищах і використовуючи двухфакторну аутентифікацію (2FA), щоб обмежити загальний ризик.

Незважаючи на те, що були розроблені більш складні атаки, фішингові зусилля залишаються успішним вектором для зловмисників. Людський фактор є найслабшою ланкою в будь-якому ланцюжку безпеки. Якщо люди скачують вкладення, клацають по посиланнях або надають свої облікові дані, хакери можуть отримати доступ до облікового запису співробітника і почати горизонтальне переміщення по мережах. Регулярне моделювання фішингу в поєднанні з кращими в своєму класі інструментами для виявлення спаму і карантину може допомогти запобігти зачіпку персоналу.

Щоб бути ефективною, політика безпеки повинна застосовуватися до всіх мережних служб і всіх користувачів мережі. Однак, з огляду на постійно зростаючу природу розгортання, політиці легко прийняти вид клаптикового пристрою, що, в свою чергу, ставить під загрозу компанію. Одним з варіантів тут є реалізація рішень для управління ідентифікацією та доступом (IAM), які керують безпекою і доступом для кожного користувача, а не покладаються на специфікації пристрою або мережі. Це безпосередньо пов'язує політику безпеки з приватними особами, спрощуючи IT-відділам настройку дозволів в міру необхідності і реалізацію змін в мережі на вимогу.

Якщо дані на сервер не зашифровані - це становить потенційну загрозу. Зловмисники знають про тенденції до мобільного і віддаленого доступу і досягають помітного успіху підслуховування небезпечних підключень або створення підроблених мереж Wi-Fi, які збирають дані для входу користувача. Для захисту своїх мереж використовуйте найнадійніше шифрування, доступне для захисту даних в дорозі і в стані спокою.

Оскільки корпоративні мережі потребують постійного збільшення, з'являється обмежена видимість для фахівців з безпеки і збільшені можливості для потенційних хакерів. Тут знання - сила: компаніям потрібні інструменти моніторингу та аналізу, які надають дані про поточний процес, сховище і

використанні системи, а також можуть оцінити середовища, щоб знайти служби або програми, які не використовуються активно.

Жодне серверне середовище не ідеальне. Програмне забезпечення може містити відомі уразливості, або зловмисники можуть виявити експлойти нульового дня. У обладнання можуть бути жорстко запрограмовані недоліки, а інструменти доступу до бази даних можуть містити невиявлені експлойти. Регулярне тестування має вирішальне значення. По-перше, компанії повинні перевірити на ризик - які служби становлять найбільшу небезпеку, не скомпрометовані чи не пошкоджені. Потім вони повинні піти в атаку, використовуючи власний талант або зовнішню допомогу, щоб безпечно зламати свою мережу і виявити критичні недоліки.

Неосвічені співробітники представляють величезний ризик. Хоча вони, ймовірно, знайомі з основами безпеки, з безпечним використанням соціальних мереж і поштових служб, багато хто з них не знайомий з корпоративними системами. Регулярне навчання, яке підкреслює загальну відповідальність за безпеку і надає реальні сценарії, може значно знизити ризик для компанії.

Ось деякі з основних загроз для web-застосунків:

XSS (або ще міжсайтовий скриптинг) ін’єкції [5]. — досить поширена вразливість, яка зустрічається у багатьох застосунках. Головна ідея XSS в тому, що зловмиснику вдається додати на сторінку JavaScript-код, якого до цього не було. Цей код буде виконуватися щоразу, коли жертви (тобто користувачі) заходитимуть на сторінку застосунку, де цей код додав зловмисник.

XSS вразливість виникає при генерації HTML-сторінки, коли розробнику потрібно помістити туди дані, вказані користувачем. Якщо хакер зможе розмістити довільний HTML-код, то отримає доступ до змін та керування відображенням вебсторінки з правами самого сайту.

Типи цих вразливостей:

1. Stored (збережена) XSS ін’єкція виникає, якщо шкідливий сценарій постійно зберігається на цільовому сервері. Сторонній код, який впроваджено в сеанс

користувача, знаходиться на вебсервері та чекає на відвідування користувача. Коли користувач запитує дані, що зберігаються в базі даних, програма може надіслати шкідливий сценарій користувачеві.

2. Reflected (відображений) XSS виникає, коли вебпрограма приймає вхідні дані від користувача, а потім негайно повертає ці дані користувачеві в небезпечний спосіб.

Наслідки від XSS ін'єкції можуть бути критичними. Так, дана ін'єкція не впливає безпосередньо на застосунок, вона спрямована на користувача. Проте зловмисник може вкрасти авторизаційні cookie користувача застосунку і тим самим його сесію. Також може перенаправити користувача на власний сайт, де розмістить форми вводу, схожі на оригінальні. І якщо користувач введе свої дані, то вони опиняться у зловмисника. Можливість для безперешкодного експлуатування XSS ін'єкції у застосунку може призвести до витоку даних та суттєвих репутаційних втрат для компанії.

Оскільки впроваджений код надходить в браузер з сайту, він є довіреним і може виконувати такі дії, як відправка авторизаційного файлу cookie користувача зловмисникові. Коли у зловмисника є файл cookie, він може увійти на сайт, нібито б він був користувачем, і зробити все, що може користувач, наприклад, отримати доступ до даних кредитної картки, переглянути контактні дані або змінити пароль.

SQL-injection – це атака з використаннями зловмисних запитів до бази даних сайту на основі мови SQL, які можуть викликати помилки конфігурації, відмову в обслуговуванні, несанкціонований доступ до внутрішньої, службової інформації (HTTP SQLi, UNION SQLi), здійснити злам бази даних MySQL (Blind SQLi) і тим самим зламати сайт, вбудувати зловмисний код (adware, spyware), змінити зовнішній вигляд/інформацію на сайті (Deface), викрасти або знищити користувацькі дані;

Успішна ін’єкційна атака може підробити посвідчення, створити нові посвідчення з правами адміністратора, отримати доступ до всіх даних на сервері або знищити / змінити дані, щоб зробити їх непридатними для використання.

Підробка міжсайтових запитів (Cross-Site Request Forgery – CSRF) дозволяють зловмиснику виконувати дії, 139 використовуючи облікові дані іншого користувача, без його відома або згоди.

Інші поширені атаки / уразливості включають: Clickjacking. У цій атаці зловмисник перехоплює кліки, призначені для видимого сайту верхнього рівня, і направляє їх на приховану нижче сторінку. Цей метод можна використовувати, наприклад, для відображення законного сайту банку, але захоплення облікових даних для входу в невидимий , контрольований зловмисником.

PHP-injection [6].– це атака на сайт з використанням PHP-коду, експлуатація помилок у вихідному коді PHP-об’єктів, блоків, форм, елементів сайту (Object Injection). Атака з використання команд та функцій PHP (Command Injection);

HTML-injection – це атака з експлуатацією вразливостей в HTML-коді сайту, а саме у HTML-тегах. Може спричинити підміну або перехоплення даних;

CSRF (Cross Site Request Forgery) – це атака з використанням міжсайтової підробки запитів, які виконують певні несанкціоновані дії на сайті від імені його користувачів, наприклад: зміна даних в профілі, зміна паролю, переказ коштів, здійснення покупки, оформлення підписки та інші шахрайські операції;

XML External Entities (XXE) – це атака на XML-файли і документи сайту, що призводить до розкриття конфіденційної, службової інформації, а також більш складних атак: XSS, CSRF, RCE.

SSRF (Server Side Request Forgery) – це атака з використанням підробки запитів на стороні сервера, коли сервер не перевіряє вхідні дані, а зловмисник зловживає довірою і виконує від його імені, користуючись загальнодоступними функціями, зловмисні запити;

CRLF-injection – це атака з використанням маніпуляції символами “Повернення каретки” і “Переведення рядка” (Carriage Return and Line Feed), які можуть використовуватися у HTTP-заголовках сайту. Таким чином можна спричинити системний збій, відправити шкідливий запит у логи сервера (Log Injection), виконати потенційно небезпечні дії збоку сервера, знищити дані;

LFI (Local File Inlcusion) – це атака, яка дозволяє зловмиснику експлуатувати вразливості сайту і переглядати внутрішні файли, обходити каталоги, включати локальні файли у URL-посилання або форми, виконувати з ними різноманітні маніпуляції;

RFI (Remote File Inclusion) – це атака з використанням віддалених файлів для виконання зловмисних операцій на сайті через вразливі компоненти сайту: URL, форми, код, API. Це призводить до віддаленого виконання шкідливого коду Remote Code Execution (RCE). Зловмисник таким чином розгортає шелли, руткіти, бекдори, підвищує привілеї і бере під контроль увесь сервер;

BruteForce – це атака з перебором даних авторизації (логіну та паролю) облікового запису, або будь-яких інших з метою отримання несанкціонованого доступу;

Spamming – це атака на сайт з використанням каналів публікації на сайті або електронної пошти з відправкою великої кількості повідомлень. У результаті велика кількість спаму може спричинити перенавантаження, переповнення жорстких дисків і оперативної пам’яті, привести до збою системних служб. Зловмисник також може з допомогою спаму проводити фішингові атаки з використанням соціальної інженерії;

Clickjakcing – це атака з використанням вразливості, яка дозволяє зловмиснику включати вміст інших сайтів на своїй фішинговій сторінці через iframe, проводити з ними різноманітні маніпуляції;

DNS-attacks – це атаки на DNS-сервер сайту з використанням вразливостей мережевих сервісів. Зловмисник може отримати записи DNS-зони (Zone Transfer

Attack), провести брутфорс піддоменів, дослідити структуру мережі, підмінити IP-адресу сервера та багато іншого.

DOS/DDOS [7]. (Denial of Service/Distributed Denial of Service) – зазвичай досягається за рахунок наповнення цільового сайту підробленими запитами, так що доступ до сайту порушується для законних користувачів. Запити можуть бути просто численними або окремо споживати великі обсяги ресурсів (наприклад, повільне читання або завантаження великих файлів). Обхід каталогів (файл і розкриття).

У цій атаці зловмисник намагається отримати доступ до частин файлової системи веб-сервера, до яких у нього не повинно бути доступу. Включення файлу. У цій атаці користувач може вказати «ненавмисний» файл для відображення або виконання в даних, переданих на сервер. Впровадження команд. Атаки з впровадженням команд дозволяють зловмиснику виконувати довільні системні команди в операційній системі сервера.

7 найпоширеніших типів кібератак на web-застосунки у 2023 році:

1. атака з використанням SQLін’єкції,

2. атака міжсайтового скриптинга (XSS),

3. атака типу відмова в обслуговуванні (DoS) / розподілена відмова в обслуговуванні (DDoS),

4. атака впровадження команд ОС,

5. атака з використанням LDAP ін’єкції (Lightweight Directory Access Protocol – Полегшений протокол доступу до каталогів),

6. атака грубої сили (brute force attack),

7. атака нульового дня.

Висновки до першого розділу

Проаналізовано, що вебсервіс - це програмна система, що ідентифікується URI, і публічні інтерфейси та прив'язки якої визначені та описані мовою XML. Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису з використанням повідомлень, що базуються на XML та передаються за допомогою інтернет-протоколів.

Досліджено, що безпека web-сервісів є критично важливим аспектом, оскільки вони забезпечують стабільну роботу всіх веб додатків та веб сайтів організації.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ТА ЗАСОБІВ БОРОТЬБИ З ВРАЗЛИВІСТЮ WEB-СЕРВІСІВ

2.1 Дослідження особливостей виявлення вразливостей web-сервісів

Захист сайту [8] — це комплексні заходи з кібербезпеки, які передбачають безпеку усіх технічних вузлів, ресурсів, конфігурацій сайту (файли, директорії,

сервери, піддомени, бази даних), з метою зміцнення, покращення, налагодження системи кіберзахисту, щоб унеможливити несанкціонований доступ, перехоплення, злам, шахрайські дії та інші зловмисні маніпуляції.

Не тільки компанії та фахівців з кібербезпеки прагнуть покращити захист власних додатків. Після популяризації випадків кібератак почали активно розвиватися мережеві безпекові рішення.

Одним з перших рішень були фаєрволи — це мережеві фільтри між корпоративною і зовнішньою (Інтернет) мережею. Ці фільтри були здатні блокувати лише підозрілі мережеві пакети на мережевому і канальному рівнях аналізуючи IP-адресу джерела та призначення, міток фрагментації й номери портів.

Сучасніші системи — такі як IPS / IDS (система запобігання вторгнень / система виявлення вторгнень) — здатні аналізувати вміст мережевих пакетів і порівнювати цей вміст з сигнатурами відомих атак. Проте більшість атак сьогодні — це експлуатація вразливостей самого застосунку, а не вразливостей фреймворків та бібліотек. Зі збільшенням кількості вебдодатків з'явилася ще одна проблема — сукупне число наявних вразливостей набагато більше, ніж кількість сигнатур в базах сучасних IPS — систем.

Для надійного захисту застосунку необхідний кардинально інший підхід: з глибоким аналізом змісту пакетів і хорошим знанням структури вебзастосунків, включаючи URL-параметри, хедери, форми введення даних і ін. Таким умовам задовольняє Web Firewall Application — брандмауер для застосунків, які здійснюють передачу даних через HTTP і HTTPS.

WAF здатний визначати [9] модель нормального функціонування програми на основі аналізу мережевого трафіку і системних журналів і її базі детектувати відхилення в поведінку програмної системи. Зокрема, WAF може виявляти атаки з використанням автоматичних засобів. При всіх перевагах WAF не покриває автоматичну перевірку усіх вразливостей застосунку. Він не може детектити

вразливості, в основі яких порушено логіки застосунку. Крім цього, існує багато способів обходу WAF, що дозволяє зловмиснику продовжувати атаку на систему.

Результатом пошуку рішення, як ефективно запобігати новішим типам кібератак стало PEN-тестування. Якщо говорити узагальнено, то в основі цього методу проста гіпотеза, щоб ефективно вдосконалювати системи та способи захисту продуктів від кібератак, необхідно розуміти, яким чином шукають ту чи іншу вразливість, і як вона влаштована зсередини. Іншими словами — щоб захиститися від хакерської атаки, потрібно усвідомлювати, за якими сценаріями вона може відбутися й де шукати головні слабкості системи. Саме завдяки активному аналізу системи на вміст дефектів коду чи в площині безпекових рішень можна виявити потенційні вразливості. Важливо, що PEN-тестування завжди проводиться з позиції потенційного нападника і може включати активне використання вразливостей. Тобто це контрольоване проникнення в систему. Про деякі з потенційних вразливостей ми поговоримо нижче.

2.2 Методи й способи боротьби зі стороннім втручанням до web-сервісу

Захист веб-сервісів – це комплекс заходів, спрямованих на запобігання атакам, витоку даних та забезпечення безперебійної роботи. З розвитком технологій та зростанням кількості веб-сервісів збільшується і кількість кібератак. Веб-програми залишаються однією з найуразливіших точок для зловмисників, оскільки часто містять чутливі дані та надають доступ до різних функцій системи.

У ході розвитку веб-безпеки [10] та поширення мобільних пристроїв сертифікат SSL став необхідний для будь-якого сайту. Деякі браузери вже мають режими, в яких сайти без SSL не відображаються взагалі. Справа йде до того, що робота таких сайтів буде припинена.

Сертифікат SSL забезпечує конфіденційність інформації та дозволяє упевнитися в достовірності сайту. Це особливо важливо при використанні відкритих бездротових мереж. Банківські картки, логіни, паролі та інші особисті дані, які захищені SSL-сертифікатом, приховані від чужих очей. Перевірити

наявність сертифіката на сайті легко, натиснувши на картинку замочка поруч з адресою сайту в рядку браузера.

Крім цього, SSL-сертифікат має значення для SEO (пошукової оптимізації). Наприклад, Google в пошуковій видачі дає перевагу сайтам з наявністю такого сертифіката.

Цікаво, що назва “SSL” використовується за інерцією. Протокол SSL, назва якого використовується для сертифікатів, застарів і є небезпечним. Безпечніше сімейство протоколів, яке прийшло на зміну SSL, називається TLS.

Важливо забезпечити використання користувачами та адміністраторами сайту складних унікальних паролів. Це один з кращих і найпростіших способів захистити облікові записи від злому і закриту інформацію від крадіжок.

Короткий посібник від Google містить основні рекомендації з безпеки паролів. Наявність великої літери та цифр у паролі, довжина якого не менше 12 символів, безумовно ускладнять крадіжку даних. При цьому можна порадити не використовувати персональну інформацію при створенні пароля, так як вона часто є загальнодоступною. Також варто рекомендувати користувачам не використовувати один і той самий пароль для різних облікових записів.

Крім усього, з боку сайту вкрай важливо не зберігати паролі у відкритому вигляді в базі даних, а використовувати відомі бібліотеки та функції для хешування паролів з додаванням “солі”.

Крім пароля, варто використовувати двофакторну аутентифікацію (2FA) або двокрокову верифікацію (2SV) як додатковий рубіж захисту у випадках витоку паролів. Якщо ви запровадите 2FA / 2SV, то для входу в обліковий запис користувачеві необхідно буде ввести підтвердження у вигляді одноразового коду з мобільного застосунку (переважно) або SMS (допустимо, хоча менш безпечно). Ця функція значно підвищить рівень безпеки, ускладнивши процес крадіжки облікового запису.

Веб-уразливості охоплюють широкий спектр недоліків безпеки, якими можуть скористатися зловмисники, щоб порушити цілісність, конфіденційність або доступність веб-додатків. Поширені типи кібератак включають міжсайтовий сценарій (XSS), коли зловмисники вставляють шкідливі сценарії у веб-вміст; підробка міжсайтового запиту (CSRF), яка передбачає несанкціоновані дії, що виконуються від імені авторизованих користувачів;

SQL-ін'єкція, що дозволяє зловмисникам маніпулювати запитами до бази даних; неправильні конфігурації безпеки внаслідок неправильно налаштованих параметрів; і витік конфіденційних даних, який може розкрити конфіденційну інформацію.

Уразливості IDOR можуть призвести до несанкціонованого доступу до конфіденційних об'єктів. Потенційні вразливості включають заголовки безпеки, завантаження файлів і клікджекінг, а також несанкціоновані перенаправлення та перенаправлення, які можуть становити ризик перенаправлення користувачів на шкідливі сайти.

Автентифікація та керування сеансами є іншими областями, схильними до вразливостей і вразливості до атак DoS і DDoS, спрямованих на припинення роботи служб. Забезпечення безпеки API є важливим для ефективної роботи. Забезпечення надійного захисту від вразливостей вимагає комплексної стратегії, яка включає безпечне кодування, регулярний аудит, навчання користувачів і впровадження технологій безпеки.

Уразливості XSS часто виникають під час розробки веб-додатків через низку типових помилок. Однією з таких помилок є недостатня перевірка введених даних, коли розробники не можуть належним чином перевірити введені користувачем дані перед відображенням їх на веб-сторінках. Цей аспект дозволяє зловмисникам вводити шкідливі сценарії, які потім виконуються браузерами користувачів, які нічого не підозрюють. Крім того, недбале кодування вихідних даних сприяє уразливостям XSS, коли розробники нехтують належним чином кодувати динамічний вміст перед його відображенням у браузері. Іншою

поширеною помилкою є використання небезпечних функцій JavaScript, особливо тих, які пов’язані з маніпулюванням моделлю об’єктів документа (DOM).

Розробники іноді покладаються на незахищені API або ігнорують впровадження заголовків безпеки, таким чином дозволяючи зловмисникам використовувати доступ браузера до певних функцій. Крім того, неправильне поводження з користувацьким вмістом, наприклад неправильне фільтрування або перевірка в коментарях користувачів або полях введення, може призвести до вразливості XSS. Недостатня обізнаність і освіта розробників щодо ризиків, пов’язаних із XSS, також є суттєвим фактором. Відсутність актуальної інформації про найновіші найкращі практики безпеки, нехтування заголовками безпеки, такими як політика безпеки вмісту (CSP), і недооцінка важливості принципів безпечного кодування – все це сприяє поширенню вразливостей XSS у веб-додатках.

Першим методом, який захищає користувачів від цього типу атак, який здійснюється на найнижчому рівні, рівні введення даних, є перевірка вхідних даних. Послідовність у застосуванні цієї практики може допомогти захистити веб-додатки від потенційних загроз.

Процес передбачає систематичну перевірку та дезінфекцію введених користувачем даних на сервері, щоб переконатися, що вони відповідають очікуваним форматам і не містять потенційно шкідливого коду. Перевіряючи введені дані, розробники можуть запобігти впровадженню шкідливих сценаріїв у веб-вміст. Ця дія захищає від використання вразливостей, які можуть поставити під загрозу безпеку браузерів користувачів. Ця процедура вимагає суворого дотримання попередньо визначених критеріїв введення, відкидаючи будь-які дані, які відхиляються від очікуваних шаблонів. Зведення до мінімуму ризику ін’єкції зловмисного сценарію у веб-програми також передбачає кодування вихідних даних.

Цей процес перетворює створений користувачами вміст у формат, який не дозволяє браузеру інтерпретувати його як виконуваний код. Нейтралізуючи спеціальні символи, які можуть бути частиною сценарію, вихідне кодування

гарантує, що введені користувачем дані розглядаються як звичайний текст, а не як частина коду, який потрібно виконати. Цей запобіжний захід є життєво важливим для запобігання атакам XSS і захисту користувачів від потенційної шкоди. Він працює в поєднанні з іншими заходами безпеки, включаючи перевірку даних, для подальшого захисту веб-додатків від кіберзагроз, що постійно змінюються. Іншим важливим заходом безпеки, який ефективно знижує ризик атак міжсайтових сценаріїв, є політика безпеки вмісту (CSP).

CSP дозволяє веб-розробникам контролювати виконання сценаріїв на веб-сторінках за допомогою повного набору директив. Це досягається за допомогою заголовків HTTP, які дозволяють вказувати надійні джерела вмісту, сценарії, таблиці стилів та інші ресурси. Спеціально вказуючи дозволені домени та типи вмісту, CSP запобігає запуску браузером сценаріїв із неавторизованих джерел, фактично зупиняючи впровадження шкідливих сценаріїв.

Ця потужна стратегія безпеки дотримується принципу найменших привілеїв, ефективне зменшення потенційного впливу вразливостей XSS. Заголовки CSP надають розробникам детальний контроль над політикою завантаження вмісту, дозволяючи м для вказівки різних директив. Ці директиви включають 'default-src', 'script-src' і 'style-src'. Крім того, CSP підтримує атрибути «nonce» і «hash», які дозволяють розробникам створювати динамічні політики, які підвищують безпеку. Правильно налаштовані заголовки Content Security Policy (CSP) не тільки допомагають запобігти атакам CrossSite Scripting (XSS), але й забезпечують комплексну стратегію безпеки, доповнюючи інші заходи безпеки, такі як перевірка вхідних даних і кодування вихідних даних.

Використання зовнішніх файлів сценаріїв і безпечних API для динамічного вмісту, а не вбудованого JavaScript, також має велике значення. Часті оновлення та технічне обслуговування програмних компонентів, включаючи бібліотеки та фреймворки, відіграють важливу роль у зменшенні кількості вразливостей. Розробники повинні бути в курсі найновіших практик безпеки, постійно навчатися та виконувати комплексні перевірки безпеки, які включають такі методи, як тестування на проникнення. Впровадження безпечного керування

сеансами, використання лише HTTP-файлів і файлів cookie Secure18, а також ретельний моніторинг і журналювання можуть допомогти захистити програму від загроз XSS.

Уразливості CSRF. Помилки під час розробки веб-додатків можуть призвести до підробки міжсайтових запитів (CSRF), що ненавмисно порушує безпеку користувача. Однією з таких помилок є неадекватна реалізація заходів захисту CSRF, наприклад відсутність генерації та перевірки унікальних маркерів CSRF для кожного сеансу користувача або невключення цих маркерів у критичні форми та запити, що потенційно створює вразливості. Розробники можуть не помічати важливості впровадження політики однакового походження, яка дозволяє зловмисникам створювати запити, які маніпулюють довірою між браузером користувача та певним сайтом. Крім того, ще однією поширеною вразливістю є нерегулярна або недостатня перевірка введених користувачами даних, особливо в ситуаціях, пов’язаних із конфіденційними транзакціями, такими як переказ коштів або оновлення паролів. Покладаючись виключно на механізми безпеки на стороні клієнта без надійної перевірки на стороні сервера, можна зробити системи вразливими до атак CSRF.

Нехтування методами безпечного кодування, включаючи неправильну автентифікацію користувача, може ненавмисно дозволити зловмисникам вжити заходів від імені авторизованих користувачів. Недостатня обізнаність і знання про ризики CSRF серед користувачів може призвести до ненавмисних дій CSRF через соціальну інженерію. Щоб посилити захист від атак CSRF, розробникам потрібно зробити більше, ніж просто використовувати маркери Anti-CSRF. Їм необхідно створити комплексну систему безпеки, яка включає постійний моніторинг і механізми адаптивного реагування. Як приклад розглянемо банківську програму, яка включає надійну систему відстеження взаємодії користувача.

Система реєструє стандартні транзакції та використовує 19 алгоритмів виявлення аномалій для виявлення незвичних моделей поведінки користувачів. Це дозволяє системі виявляти потенційні CSRF-атаки в реальному часі. У разі ймовірного інциденту CSRF швидко впроваджуються процедури реагування на

інцидент, щоб зменшити загрозу, звести до мінімуму будь-які перешкоди для користувачів і захистити конфіденційність конфіденційної фінансової інформації. Активна участь у спільноті кібербезпеки є цінним джерелом знань та ідей.

Розробники, які відвідують конференції з безпеки, беруть участь у форумах і роблять внески в платформи аналізу загроз, отримують колективне розуміння нових векторів атак CSRF. Наприклад, дискусії в спільноті можуть виявити нові тактики, які використовують зловмисники, як-от використання методів соціальної інженерії для маніпулювання користувачами, щоб вони несвідомо виконували дії CSRF. Розробники можуть використовувати цю колективну обізнаність, щоб адаптувати свої стратегії захисту, закриваючи потенційні вразливості до того, як їх використають. Навчаючи користувачів, дуже важливо використовувати відповідні приклади для пояснення концепції атак CSRF. Розглянемо сценарій, коли користувач входить на відому платформу електронної комерції та несвідомо ініціює фінансову операцію на іншому веб-сайті за допомогою маніпулятивного посилання.

Підвищуючи обізнаність про такі ризики та наголошуючи на важливості бути пильними в Інтернеті, користувачі стають більш вправними у виявленні підозрілої активності. Цей підхід, орієнтований на клієнта, у поєднанні із запобіжними засобами, такими як токени Anti-CSRF, створює потужний дворівневий захист від ризиків CSRF. Таким чином, використання токенів Anti-CSRF є важливим елементом онлайн-безпеки для. Незважаючи на ефективність, ще ефективніше інтегрувати їх у комплексну інфраструктуру безпеки. Розробникам слід користуватися перевагами безперервного моніторингу, взаємодіяти зі спільнотою безпеки та навчати користувачів, як запроваджувати численні засоби захисту CSRF.

Такий підхід не тільки захищає від поточних загроз, але й дозволяє додаткам адаптуватися до нових викликів кібербезпеки, забезпечуючи безпечну роботу в Інтернеті для користувачів. SQL Injections Пов’язані з SQL уразливості залишаються значною та широко поширеною загрозою для веб-додатків. Ці вразливості часто виникають через повторювані помилки як під час розробки, так

і під час d етапи обслуговування продукту. Однією з найпоширеніших помилок є нездатність належним чином перевірити та очистити введені користувачем дані. Недостатня перевірка введених користувачем даних створює можливість для зловмисників вводити шкідливі SQL-запити в програму.

Введені запити можуть маніпулювати базою даних, потенційно призводячи до несанкціонованого доступу та витоку або втрати конфіденційної інформації. Динамічні запити SQL без належної параметризації також є поширеною помилкою. Конкатенація користувацького введення в інструкції SQL дозволяє зловмисникам маніпулювати структурою запиту, полегшуючи доступ або маніпулювання даними без авторизації. Неправильна обробка помилок є ще однією вразливістю, яка спричиняє вразливості SQL.

Розкриття складних повідомлень про помилки користувачам у виробництві ненавмисно розкриває структуру бази даних, надаючи зловмисникам цінну інформацію для створення цільових атак SQL-ін’єкцій. Крім того, слабкі засоби контролю доступу та неправильне керування привілеями можуть призвести до надання обліковим записам програми надмірних дозволів. Це дозволяє зловмисникам використовувати ці привілеї, щоб отримати неавторизований доступ або маніпулювати даними, тим самим ставлячи під загрозу загальну безпеку.

Ефективне вирішення питань безпеки SQL потребує ретельного підходу, який включає впровадження методів безпечного кодування, проведення періодичних оцінок безпеки, створення протоколів обробки помилок, забезпечення контролю доступу та послідовність у використанні конкретних термінів, скорочень і символів. Важливо використовувати точну мову, коли цього вимагає ситуація, зменшити складність і уникати надмірностей або фраз-заповнювачів, підтримувати нейтральний тон і дотримуватися стандартних умов мови для досягнення ясності, зв’язності та граматичної правильності. Впровадження цих заходів може підвищити стійкість веб-додатків до постійно мінливої загрози вразливостей SQL, захистити конфіденційні дані та зберегти цілісність додатків і баз даних. Параметризовані запити або підготовлені оператори також

використовуються для захисту від атак SQL-ін’єкцій. Ці методи дозволяють розробникам відокремлювати введення користувача від запитів SQL, створюючи захисний бар’єр, який запобігає впровадженню шкідливого коду. Послідовність і дотримання стандартизованої мови та точний вибір слів є критично важливими для реалізації цих заходів безпеки. Крім того, формальний тон із наголосом на однозначній мові та логічній послідовній структурі допоможе забезпечити ясність і послідовність документації. Використовуючи параметри як заповнювачі для введення користувача, введення розглядається виключно як дані, а не як виконуваний код.

Цей підхід підвищує безпеку додатків, запобігаючи маніпулюванню операціями з базою даних за допомогою введених SQL-запитів. Надійна реалізація перевірки вхідних даних є важливим механізмом безпеки, який гарантує, що вхідні дані відповідають очікуваним форматам, включаючи перевірку довжини, формату та діапазону. Базової перевірки типу даних недостатньо. Ретельна перевірка даних користувача на відповідність заздалегідь визначеним критеріям значно мінімізує ризики впровадження SQL для розробників. Ця практика підвищує безпеку, відмовостійкість і стійкість додатків. Дотримання принципу найменших привілеїв є потужним методом захисту для зменшення потенційного впливу атак SQL-ін’єкцій. Цей принцип вимагає, щоб облікові записи бази даних, пов’язані з програмою, отримували лише мінімальний рівень доступу, необхідний для виконання призначених завдань.

Обмеження привілеїв таких облікових записів зменшує потенційну шкоду, яку може завдати успішне впровадження SQL, і, як наслідок, робить середовище бази даних більш безпечним. Регулярні перевірки безпеки та оцінки вразливостей необхідні для виявлення й усунення потенційних уразливостей SQL-ін’єкцій. Поєднання автоматизованих інструментів і ручного тестування має вирішальне значення для виявлення слабких місць у коді вашої програми та її взаємодії з базою даних. Такі аудити служать не лише профілактичним заходом, але й допомагають вам постійно вдосконалювати протоколи та методи безпеки.

Установивши брандмауер веб-додатків (WAF), ви можете створити додатковий рівень захисту від атак SQL-ін’єкцій. WAF ретельно перевіряє вхідний трафік і виявляє шаблони, які відповідають розпізнаним векторам атак SQL-ін’єкції. Проактивно виявляючи та запобігаючи зловмисним спробам, WAF діє як динамічний щит, значно підвищуючи загальну безпеку веб-додатків. Іншим важливим аспектом безпеки є безперервне навчання кодуванню, яке активно розвиває культуру безпеки. Висвітлюючи ризики впровадження SQL і навчаючи розробників найновішим заходам безпеки, вони отримують необхідні знання та навички, щоб вони могли ефективно підтримувати надійні протоколи безпеки. Навчання підвищує обізнаність щодо безпеки та дає змогу розробникам протистояти загрозам впровадження SQL, забезпечуючи надійні заходи безпеки для своїх програм. ClickJacking

Clickjacki ng вразливості, поширена загроза веб-безпеці, коли зловмисник обманом змушує користувача натиснути прихований або невидимий елемент, що призводить до небажаних дій, може посилюватися різними помилками веб-розробки. Однією з поширених помилок є недостатнє впровадження методів обходу рамок.

Розробники можуть використовувати методи запобігання фреймування своїх веб-сторінок іншими сайтами, але такі заходи часто є неповними або неправильно налаштованими, що призводить до вразливостей. Іншою поширеною помилкою є відсутність відповідних заголовків безпеки, таких як X-Frame-Options, які пояснюють, як веб-сторінку слід вставляти у фрейми.

Ігнорування цих заголовків або їх неправильне застосування наражає веб-сторінки на клікджекінг. Покладатися виключно на засоби безпеки на стороні клієнта, наприклад рішення на основі JavaScript, без їх резервного копіювання засобами безпеки на стороні сервера може бути критичною помилкою. Розробники можуть недооцінити важливість перевірки на стороні сервера і натомість зосередитися на сценаріях на стороні клієнта, залишаючи свої програми вразливими до спроб клікджекінгу, які обходять елементи керування на стороні клієнта. Не недооцінюйте важливість регулярного аудиту та оновлення безпеки. З

розвитком веб-технологій можуть з’явитися вразливі місця в безпеці, а якщо не підтримувати компоненти програмного забезпечення, такі як фреймворки та бібліотеки, в актуальному стані, ваші веб-додатки можуть стати вразливими для клікджекінгу. Впровадження політики безпеки вмісту (CSP) використовується для контролю джерел, з яких вміст можна завантажувати на ваш веб-сайт.

Добре розроблений CSP визначає авторизовані домени, які можуть вбудовувати веб-програми, тим самим зменшуючи ризик клікджекінгу, обмежуючи місця, де вміст може відображатися у фреймі. Ця політика діє як надійний запобіжний захід проти різних типів веб-атак, тим самим підвищуючи загальну безпеку програми. Вбудовування сценаріїв, які запобігають відображенню продукту у фреймі на веб-сторінках, є ефективним методом запобігання спробам клікджекінгу. Зазвичай написані на JavaScript, ці сценарії активно визначають, чи сторінка у фреймі знаходиться на іншому сайті, і виконують коригувальну дію, щоб вийти з фрейму. Забезпечуючи відображення вмісту лише в призначеному для нього контексті, сценарії, що розбивають кадри, забезпечують додатковий рівень захисту від злому кліків, тим самим підвищуючи безпеку веб-програми. Візуальні індикатори відіграють не менш важливу роль у підвищенні обізнаності користувачів і полегшенні ідентифікації законного вмісту.

Відображаючи на помітному місці водяні знаки або логотипи, які зловмисникам важко відтворити, користувачі можуть перевірити автентичність веб-сторінки. Ці візуальні підказки дозволяють користувачам приймати обґрунтовані рішення щодо достовірності вмісту, з яким вони взаємодіють, що сприяє підвищенню безпеки користувачів. Інформування користувачів про ризики, пов’язані з клікджекінгом, є фундаментальним аспектом комплексної стратегії безпеки. Підвищуючи обізнаність про потенційні загрози та наголошуючи на важливості перевірки легітимності веб-сайтів, користувачі стають активними учасниками захисту від клікджекінгу. Поінформовані користувачі, швидше за все, будуть проявляти обережність, розпізнавати підозрілу активність і повідомляти про можливі інциденти безпеки, тим самим сприяючи покращенню веб-безпеки.

Ще один метод, який повинен працювати в поєднанні з попередніми, це впровадження реферерів. Він використовується для регулювання розкриття інформації в заголовку реферера HTTP. Цей підхід допомагає запобігти розкриттю конфіденційних даних для потенційно зловмисних сайтів, які можуть здійснювати атаки зловмисників. Ретельно налаштована політика реферерів включає додатковий захист, який мінімізує можливість ненавмисного розкриття інформації під час взаємодії з користувачем. Важливо посилити інфраструктуру безпеки за допомогою брандмауера веб-додатків (WAF), який має спеціалізовані можливості для виявлення та запобігання клікджекінгу. WAF діє як проактивний захист, аналізуючи вхідний трафік, виявляючи блоки, що вказують на спроби клікджекінгу, і фільтруючи потенційно зловмисні запити. Інтеграція WAF в архітектуру безпеки забезпечує додаткову лінію захисту, особливо в динамічному онлайн-середовищі.

2.3 Засоби для пошуку вразливостей web-сервісів

Вибираючи інструмент для сканування веб-уразливостей, дуже важливо ретельно оцінити ваші конкретні потреби з огляду на такі фактори як складність веб-додатків, бажана глибина сканування та бюджет. Різні інструменти служать різним цілям, тому вибирати варто дуже ретельно.

Деякі інструменти спеціалізуються [11] на автоматизованому сканування на наявність поширених уразливостей, тоді як інші пропонують більш комплексні рішення, що включають можливості ручного тестування Краще вибрати інструмент, який підтримує і точно виявляє вразливості в веб-технологіях, які застосовуються в конкретному продукті. Крім того, дуже важливими є можливості звітності інструменту. Чіткі та дієві звіти є дуже корисними для ефективного виправлення ситуації. Тому дуже важливо вибрати інструмент, який надає чіткі та вичерпні звіти, беручи до уваги бюджетні обмеження.

Є безкоштовні програми з відкритим кодом, такі як OWASP ZAP та OpenVAS, які можуть бути особливо вигідні для економічних організацій, оскільки вони

пропонують потужні можливості сканування, не вимагаючи ліцензійних платежів. Важливо бути в курсі останніх подій у сфері кібербезпеки.

Слід шукати інструмент, який регулярно отримує оновлення від постачальника або спільноти для розширення бази даних уразливостей. Це допоможе гарантувати, що сканер зможе виявляти нові загрози та вразливості. Інтеграція з іншими інструментами безпеки, такими як системи управління інформацією та подіями безпеки (SIEM) або платформи відстеження проблем може значно спростити загальний процес забезпечення безпеки. Бажано вибирати інструменти, які виявляють уразливості, а також допомагають визначати пріоритети та керувати зусиллями щодо їх усунення.

Регулярне оновлення та вдосконалення інструментів сканування має істотне значення для того, щоб йти в ногу зі змінами у веб-застосунках. Адаптація конфігурацій до мінливого ландшафту загроз та конкретних потреб тій чи іншій організації має далеко ще не останній пріоритет. Регулярний перегляд стратегії сканування гарантує ефективність інструментів у виявленні та усуненні потенційних уразливостей.

Одним із таких інструментів може бути BurpSuite. Burp Suite, створений PortSwigger – це набір інструментів, спеціально розроблених для тестування. безпеки веб-застосунків. Його особливістю є потужний сканер, який автоматизує процес виявлення та оцінки уразливостей безпеки у веб-додатках. Його універсальність робить його популярним серед фахівців з кібербезпеки, оскільки він підходить як для ручного, так і для автоматизоване тестування. Burp Suite має потужний автоматичний сканер, який знаходить. найпоширеніші вразливості веб-додатків, включаючи SQL-ін'єкції, міжсайтовий скриптинг (XSS) та інші поширені вразливості. Це дозволяє швидко та ефективно виявляти можливі ризики безпеки. Тим не менш, автоматизовані сканери мають обмеження, що відрізняє Burp Suite від інших продуктів завдяки можливостям ручного тестування. Аналітики безпеки можуть взаємодіяти з веб-додатками,

досліджувати їх тонкощі і виявляти непомітні вразливості, які можуть вислизнути від автоматизованих інструментів.

Це дуже важливо для виявлення уразливостей, які потребують глибокого розуміння контексту веб-додатку. Крім того, Burp Suite дозволяє фахівцям з безпеки відтворювати різні сценарії атак та оцінювати реакцію веб-додатків на потенційні загрози. Це допомагає розпізнавати вразливості та допомагає розробникам зменшити потенційні ризики перед тим, як ними скористаються зловмисники.

Однією з основних переваг Burp Suite є його простий у використанні інтерфейс, що підходить як для початківців, так і для досвідчених фахівців з безпеки. Крім того, інструмент забезпечує безперешкодне співробітництво в командам безпеки, сприяючи обміну ідеями під час тестування. Веб-сайт PortSwigger служить офіційною платформою Burp Suite, надаючи користувачам необхідну документацію, оновлення та ресурси. Оскільки ризики кібербезпеки продовжують розвиватися, PortSwigger постійно оновлює та вдосконалює Burp Suite, щоб відповідати новим викликам безпеки та підтримки нових веб-технологій.

Іншим хорошим сканером для веб-застосунків може бути OWASP ZAP, також відомий як Open Web Application Security Project ZAP. OWASP ZAP призначений для виявлення вразливостей безпеки у веб-додатках і пропонує широкий спектр функцій, що задовольняють. користувачів із різним рівнем підготовки. Як інструмент із відкритим вихідним кодом, ZAP отримує постійні внески та удосконалення від глобальної спільноти фахівців з безпеки, пов'язаних з OWASP.

Це гарантує, що інструмент залишається актуальним у середовищі кібербезпеки, яка постійно змінюється. Автоматичні сканери ZAP є однією з його ключових переваг, оскільки вони можуть виявляти такі поширені вразливості, як SQL-ін'єкції та міжсайтовий скриптинг (XSS) Цей інструмент суттєво допомагає 28 користувачам на ранніх стадіях тестування безпеки, дозволяючи їм ефективно

розпізнавати та ранжувати потенційні загрози. Адаптивність ZAP є ще однією важливою характеристикою, оскільки пропонує ресурси, придатні як для новачків, так і для експертів. інтуїтивно зрозумілий інтерфейс полегшує тестування безпеки веб-додатків для початківців, тоді як розширені функціональні можливості дозволяють досвідченим експертам легко проводити комплексні оцінки. ZAP відрізняється своїм інтерактивним методом тестування безпеки, який дозволяє користувачам активно взаємодіяти з веб-додатками та імітувати реальні сценарії атак.

Це дозволяє їм виявляти тонкі вразливості, які автоматизовані сканери можуть не помітити, забезпечуючи ретельне покриття безпеки. Користувачі можуть легко отримати доступ до новітніх версій OWASP ZAP через платформу, яка дає їм найрелевантніші поради та способи вирішення проблем. Окрім того, доступні на сайті освітні ресурси. дають можливість користувачам поглибити свої знання про безпеку веб-додатків та отримати цінну інформацію про ефективне використання OWASP ZAP у реальних ситуаціях.

Спільнота OWASP ZAP сприяє створенню. інклюзивного середовища, де користувачі можуть обмінюватися досвідом, шукати рекомендації та вирішувати проблеми. через форуми та інші канали зв'язку, прагнучи вдосконалити та просувати OWASP ZAP вперед.

Висновки до другого розділу

Проаналізовано, що захист web-сервісів організації має важливе значення для охорони цифрових активів організації та конфіденційної інформації від різних загроз безпеці.

Досліджено, які загрози існують для web-сервісів організації, їхній влив на її функіонування та можливі засоби для їх перешкоджанню.

3 РОЗРОБКА РЕКОМЕНДАЦІЙ ТА СТВОРЕННЯ БЕЗПЕКОВИХ РІШЕНЬ ЩОДО ЗАХИСТУ WEB-СЕРВІСІВ

3.1 Створення рекомендацій для захищенності web-систем

Для захисту від WEB-атак класичним пристроєм є Web Application Firewall. Міжмережний екран веб-додатків (Web Application Firewall) застосовує набір правил захисту до протоколів високого (прикладного) рівня HTTP/HTTPS, FTP/FTPS. Класичне розміщення WAF в мережі – в режимі зворотного проксі-сервера перед захищеними веб-серверами. В набір функцій WAF зазвичай входять такі типові механізми захисту:

– валідація протоколу;

– сигнатурний аналіз;

– захист сесій та cookie;

– блокування витоку даних;

– розпізнавання атак (з негативної моделі, атаки на додатки, мережу, веб-сервер і атаки на ОС); – можливість створення власних правил захисту;

– машинне навчання.

Загальні вимоги до сучасного Web Application Firewall:

– системні компоненти WAF повинні відповідати вимогам PCI DSS;

–можливість реагування на загрози, описані в OWASP Top 10;

– інспектування запитів і відповідей відповідно до політики безпеки;

– журнал роботи подій;

– запобігання витоку даних

– інспекція відповідей сервера на наявність критичних даних;

– інспектування всього вмісту веб-сторінок, включаючи HTML, DHTML і CSS, а також протоколів доставки вмісту (HTTP / HTTPS);

–інспектування будь-якого протоколу або конструкції даних, що використовуються для передачі даних веб-додатків;

– захист від загроз, спрямованих безпосередньо на WAF;

– підтримка SSL/TLS-термінації з'єднання;

– запобігання або виявлення підробки ідентифікатора сесії;

– автоматичне оновлення сигнатур атак і застосування їх;

– підтримка пристроєм клієнтських SSL-сертифікатів;

– підтримка апаратного зберігання ключів (FIPS). З наведеного зробимо висновок про недоліки системи, а саме про нездатність WAF протистояти деяким видам атак.

Даних загальних функцій WAF недостатньо для повного захисту інтернет-ресурсів, так, як WAF не захищає від атак низького рівня, таких, як, наприклад DDoS, SYN-flood, TCP-flood, UDP-flood. Таким чином виникає потреба в додатковому захисті, а саме в створенні комплексу для покращення системи безпеки захисту веб-ресурсів.

Для покращення роботи системи безпеки веб-ресурсів необхідно створити захист від атак низького рівня. Для цього можна використати системи IPS та Firewall. Firewall буде формувати доступ до портів та забезпечувати мінімальний захист, IPS буде відстежувати атаки низького рівня та реагувати на зміни потоків трафіку. Intrusion Prevention System (система протидії вторгнень) – система, яка розпізнає ознаки вторгнення, виявляє атаки і запобігає їм. Під час аналізу використовуються різні методи виявлення атак – сигнатурний, поведінковий і ідентифікація аномалій в протоколах.

Також, всі види IPS технологій, як правило, виконують наступні функції:

– IPS зупиняє саму атаку.

– Блокує зловмисну частину, дозволяючи неураженій частині проникати до системи

– Повідомляють адміністраторів безпеки у разі важливих подій, що спостерігаються у системі

– Реагують на інциденти, змінюючи середу безпеки для зривання атаки.

– Створюють звіти. Загальні вимоги до IPS та типи подій, які найбільш часто виявляються:

– Дослідження і атаки прикладного рівня (наприклад, переповнення буфера, підбір пароля, передача шкідливих програм). Більшість мережевих IPS аналізують протоколи додатків.

– Дослідження і атаки транспортного рівня (наприклад, сканування портів, незвичайна фрагментація пакетів, SYN floods). Найбільш часто аналізуються протоколи транспортного рівня

– TCP і UDP.

– Дослідження і атаки мережевого рівня (наприклад, підміна IP-адреси, ненормальні значення заголовка IP).

– Неочікувана робота додатків (наприклад, хости виконують несанкціоновані дії).

– Порушення політики (наприклад, використання заборонених протоколів). Firewall (міжмережний екран)

– система мережної безпеки, яка відстежує і контролює вхідний і вихідний трафік на основі заздалегідь визначених правил безпеки.

Міжмережний екран, зазвичай, встановлює бар'єр між захищеною внутрішньою мережею і зовнішньою незахищеною мережею. Основною його метою є захист внутрішньої мережі або окремих її вузлів від несанкціонованого доступу. Міжмережний екран контролює доступ до ресурсів мережі за допомогою позитивної моделі управління (у внутрішню мережу потрапляє тільки дозволений правилами трафік, весь інший трафік заборонений).

Загалом міжмережні екрани діляться на дві категорії:

– Міжмережні екрани мережного рівня дозволяють чи забороняють трафік, базуючись на адресах джерела IP і адресах чи портах призначення IP.

– Міжмережні екрани прикладного рівня аналізують протоколи прикладного рівня, спостерігаючи за активністю протоколу по відношенню до визначеного профілю і дозволяють чи забороняють трафік, базуючись на відхиленнях від профілю.

Типові функції Firewall:

– Контроль доступу до вузлів в мережі

– Фільтрація доступу до незахищених служб

– Контроль порядку доступу до мережі

– Запобігає спробам доступу з зовнішньої і з внутрішньої мережі

– Перешкоджання отримання закритої інформації із внутрішньої захищеної мережі.

Таким чином, в комплексі, системи будуть захищати інтернет-ресурси на всіх рівнях моделі OSI.

WAF впроваджується в систему в режимі зворотнього проксі-сервера перед захищеними веб-серверами. IPS впроваджується в комплекс в режимі Transparent. Отримуючи запити, допущені Firewall, IPS аналізує протоколи і зупиняє певні види атак, надалі дані передаються до WAF, де обробляються і також блокуються атаки функціоналу WAF.

Відповіді веб-сервера знову повертаються до WAF, де перевіряються на наявність витоку даних. Після перевірки, дані ідуть до користувача. Використання даної комплексної системи захисту інтернет-ресурсів у складі Firewall, IPS та WAF забезпечує на 30% більшу ефективність ніж використання звичайного WAF. Використання сучасних високопродуктивних Firewall та IPS дозволить блокувати потрапляння шкідливих файлів у внутрішню захищену мережу, забезпечить додаткову безпеку і зменшить ризики цілеспрямованих атак на IT-ресурси. Даний комплекс дозволить збільшити захищеність будь-якого інтернет-ресурсу, зменшити навантаження на адміністраторів IT-систем, та забезпечити більш ефективну обробку легітимних користувачів інтернет-ресурсів.

3.2 Додаткові рекомендації для захисту від можливих загроз

Існує вичерпний перелік стратегій та методів, які дозволять уникнути виникнення вразливостей такого роду в продукті. До них належать: валідація вхідних даних, екранування даних, використання CSP заголовку в запитах, використання HTTPOnly куків та оновлення бібліотек що використовуються.

Валідація є процесом перевірки введених користувачем даних на відповідність певним критеріям. Вона допомагає переконатися, що введені дані не

містять шкідливого коду, який може бути виконаний у браузері користувача. Ескейпінг спеціальних символів є одним з методів валідації введених даних для запобігання XSS-атак. Його основна ідея полягає в тому, що спеціальні символи, які можуть використовуватись для ін'єкції шкідливого коду, заміняються спеціальними послідовностями, які не інтерпретуються браузером як код.

Нижче наведені приклади деяких спеціальних символів та їх ескейпінгу:

- < (менше) і > (більше):

Оригінальний ввід: <script> alert('XSS') </script>

Ескейпований ввід: <script>alert(`XSS`)</script>

- & (амперсанд):

Оригінальний ввід: user&admin

Ескейпований ввід: user&admin

- " (подвійні лапки):

Оригінальний ввід: <img src=’’xss.jpg``>

Ескейпований ввід: <img src="xss.jpg">

- ' (одинарні лапки):

Оригінальний ввід: alert('XSS')

Ескейпований ввід: alert('XSS')

екранування спеціальних символів:

Приклад коду на мові PHP для екранування спеціальних символів:

```
$userInput = "<script> alert('XSS') </script>";
$escapedInput = htmlspecialchars($userInput, ENT_QUOTES, 'UTF-8');
echo $escapedInput; // Виведе <script>alert('XSS');</script>
```

Заміна спеціальних символів на ескейп-послідовності гарантує, що браузер буде сприймати їх як звичайний текст, а не як код, що має бути виконаний. Це дозволяє запобігти XSS-атакам, оскільки шкідливий код буде відображатись як текстова інформація на сторінці, а не виконуватись.

Варто врахувати, що ескейпінг спеціальних символів не розв'язує всіх проблем, пов'язаних з XSS-атаками. Він може бути використаний як один з етапів захисту, тож варто поєднувати його з іншими методами, такими як валідація

формату, білий список тегів і атрибутів, а також фільтрація небезпечних кодів, для досягнення комплексного захисту від XSS-атак.

Інший метод виконує валідацію формату саме вхідних даних. Цей підхід полягає у перевірці введених користувачем даних на відповідність певному формату, що дозволяє уникнути введення потенційно шкідливого коду. –

Важливо перевірити, чи містяться введені дані HTML-теги. Для цього можна використовувати регулярні вирази або спеціальні функції для перевірки наявності < та >, які вказують на можливість введення HTMLкоду.

Приклад коду на мові JavaScript для перевірки наявності HTML-тегів:

```
function isHtmlTagsPresent(input) {
var htmlTagsRegex = /]+>/g;
return htmlTagsRegex.test(input); }
 // Приклад використання
var userInput = "<script> alert('XSS') </script>";
var containsHtmlTags = isHtmlTagsPresent(userInput);
console.log(containsHtmlTags); // Виведе true
```

- Перевірка формату даних:

Деякі типи даних мають визначений формат, який можна перевірити для запобігання введенню шкідливого коду. Наприклад, для електронних адрес можна використовувати регулярні вирази, щоб перевірити, чи має введена адреса правильний формат.

Приклад коду на мові JavaScript для перевірки формату електронної адреси:

```
// Оголошення функції
function isValidEmail(email) {
 // Оголошення змінної, значенням якої є регулярний вираз, який буде перевіряти формат введених даних
var emailRegex = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
// Повернення результатів перевірки вхідних даних
```

```
return emailRegex.test(email); }

// Приклад використання
// Оголошення змінної, значення якої потрібно перевірити
var userEmail = "example@example.com";
// Записування в змінну значення true або false в залежності від результатів функції
var isValid = isValidEmail(userEmail);
// Вивід результатів виконання функції console.log(isValid);
// Виведе true
```

Важливо враховувати особливості конкретної мови програмування або фреймворка, з яким ви працюєте, та використовувати належні методи валідації для конкретних типів даних, що вводяться користувачами.

Настуний метод валідації даних який ми розглянемо називається “білий список” (“Whitelist”). Цей підхід полягає в обмеженні типів HTML-тегів і атрибутів, які можуть бути використані у введених даних, дозволяючи лише допустимі елементи та їх атрибути. Використання білого списку забезпечує контроль над тим, які елементи і атрибути можуть бути відображені на сторінці, запобігаючи виконанню шкідливого скрипту.

У процесі валідації можна створити список допустимих HTML-тегів, які можуть бути використані в текстових полях або коментарях. Наприклад, при обробці коментарів можуть бути дозволені теги <p>, <a>, , .

Крім HTML-тегів, можна встановити білий список для атрибутів і їх значень. Наприклад, при обробці введених URL-адрес можна дозволити лише http:// або https:// як схему, виключаючи інші протоколи, які можуть бути потенційно шкідливими.

В залежності від потреб, замість відкидання недопустимих елементів можна також використовувати метод обрізання. У цьому випадку, якщо зустрічається

недопустимий елемент або атрибут, він буде видалений або замінений на безпечний еквівалент. Наприклад, якщо введений текст містить

HTTP був розроблений для обміну базовим текстом і зображеннями. Проте, оскільки веб-додатки ставали дедалі складнішими, виникла потреба в удосконаленні протоколу. Включення заголовків HTTP в HTTP/1.0 стало значним покращенням, оскільки дозволило передавати додаткову інформацію разом із запитами та відповідями. HTTP-заголовки є ключовою частиною протоколу, яка ідентифікує клієнтів і сервери, визначає мову і тип контенту, керує сеансами і впроваджує правила кешування. Вони також роблять значний внесок у безпеку та автентифікацію веб-додатків, встановлюючи правила обміну конфіденційними даними. Заголовки також відповідають за оптимізацію завантаження вебсторінок і зменшення навантаження на сервер за рахунок керування кешуванням ресурсів. Крім того, вони допомагають оптимізувати передачу даних, визначаючи кодування і розмір пакетів. Крім того, HTTP-заголовки встановлюють правила та дозволи для міждоменних запитів, що особливо важливо для сучасних веб-додатків та API, які потребують безпечного обміну даними між різними джерелами.

Для захисту від XSS потрібно використовувати заголовок CSP. Замість того, щоб покладатися виключно на сервер для санітарної обробки та перевірки користувацького контенту, CSP зміщує акцент на контроль джерел, з яких контент може бути завантажений на веб-сайт, таким чином підвищуючи безпеку. CSP дозволяє адміністраторам веб-сайтів встановлювати правила для браузера щодо завантаження вмісту. Директива script-src відіграє важливу роль у цьому процесі, вказуючи авторизовані джерела для виконання скриптів. Явно визначаючи довірені джерела, CSP ефективно зупиняє виконання скриптів з несанкціонованих або шкідливих джерел.

Зрештою, CSP надає атрибут "nonce", що дозволяє розробникам створювати окремі криптографічні nonce для кожного скрипта. Ці нонси можуть бути включені безпосередньо в теги скриптів на стороні сервера. Після цього браузер підтверджує, чи відповідає nonce в тезі скрипта nonce, зазначеному в заголовку

CSP, перед тим, як дозволити виконання скрипта. Такий незалежний від nonce підхід додає додатковий рівень безпеки, гарантуючи, що виконуються лише скрипти з дійсними nonce.

Директива CSP "strict-dynamic" дозволяє включати скрипти з довірених джерел, навіть якщо вони не вказані в заголовку CSP. Це необхідно для того, щоб дозволити завантажувати динамічний вміст, зберігаючи при цьому безпеку. Однак вбудовані скрипти всеодно вимагатимуть відповідного nonce або хешу для перевірки. Варто зазначити, що CSP допомагає знизити ризик XSS-атак, обмежуючи використання вбудованих скриптів, обробників подій і стилів. Термін "unsafe-inline" використовується для блокування вбудованих скриптів і стилів, що спонукає розробників переміщати свої скрипти і стилі в зовнішні файли. Цей метод обмежує потенційні шляхи для зловмисників, що планують впровадити шкідливий код, таким чином зменшуючи ризик атак. Також, CSP забезпечує надійний механізм звітності за допомогою директиви "report-uri" у разі порушення політики. Коли виникає порушення, браузер надсилає звіт на вказаний URI, що дозволяє адміністраторам контролювати і швидко вирішувати можливі проблеми безпеки.

Концепція файлів cookie виникла через необхідність зберігати невеликі біти даних на пристрої користувача. Ці файли, які зазвичай називають "cookies", були створені для того, щоб покращити досвід перегляду веб-сторінок і дозволити веб-сайтам ідентифікувати та нагадувати користувачам про себе.

Спочатку файли cookie були створені компанією Netscape у 1994 році і призначалися для збереження інформації про стан користувача під час перегляду ним різних сторінок веб-сайту. Файли cookie слугують декільком цілям і були розроблені для вирішення певних проблем, пов'язаних з веб-взаємодією. Однією з головних причин їх створення було полегшення управління сеансами, що дозволило б веб-сайтам розпізнавати користувачів і підтримувати їхній статус входу в систему. Це було особливо важливо для онлайн-сервісів, які потребують автентифікації користувачів, таких як платформи електронної пошти та ранні веб-сайти електронної комерції.

З розвитком онлайн-середовища файли cookie знайшли додаткові можливості використання. Вони відіграли важливу роль у персоналізації користувацького досвіду завдяки збереженню уподобань та налаштувань. Наприклад, веб-сайти почали використовувати файли cookie для збереження мовних уподобань, вибору тем та інших користувацьких конфігурацій, що призвело до створення більш персоналізованого та зручного інтерфейсу. Створення файлів cookie мало основну мету - полегшити таргетовану рекламу, відстежуючи поведінку користувачів в Інтернеті. Такий моніторинг дозволив би рекламодавцям показувати персоналізовану та релевантну рекламу. Однак ця практика викликала занепокоєння щодо конфіденційності, що призвело до створення нормативних актів та ініціатив, спрямованих на захист даних користувачів. З часом файли cookie стали більш обширними і почали включати в себе різні типи, кожен з яких виконує певну функцію і робить свій внесок у загальну функціональність інтернету.

Файли cookie стали вирішальним аспектом оптимізації користувацького досвіду. Тим не менш, виникли дебати про конфіденційність, що призвели до таких досягнень, як атрибути файлів cookie SameSite і поява альтернатив, таких як "відбитки пальців" браузерів. 38 Атрибут HTTPOnly використовується для HTTP-файлів cookie, щоб обмежити доступ до файлів cookie з JavaScript-скриптів, як захисний захід від XSS (міжсайтового скриптингу).

Коли сервер встановлює HTTPOnly для файлу cookie, це означає, що до нього не можна отримати доступ через об'єкт document.cookie в JavaScript. HTTPOnly файли cookie використовуються виключно для взаємодії з сервером при доступі до ресурсів. Встановлення HTTPOnly для файлів cookie має на меті запобігти доступу зловмисників до них у випадку, якщо XSS-уразливість вже дозволила впровадити шкідливий код на сторінку. Це мінімізує ризик втрати конфіденційної інформації, яка може зберігатися в файлах cookie.

Важливо визнати, що HTTPOnly не є комплексним рішенням і не може замінити інші заходи безпеки, включаючи точну обробку вхідних даних та інші засоби контролю безпеки. Крім того, слід вжити негайних заходів для усунення

XSS-уразливості, оскільки HTTPOnly лише пом'якшує вплив такого недоліку, а не усуває його повністю.

Для комплексного захисту від CSRF вразливостей, необхідно створити комплексну систему безпеки, яка передбачає постійний моніторинг та адаптивні механізми реагування. Ця система безпеки повинна включати в себе: використання CSRF токенів, SameSite Cookies, Double Submit Cookies та HTTP Referer Header.

Унікальний токен CSRF створюється кожного разу, коли користувач входить в систему або взаємодіє з нею. Цей маркер, який може бути випадковим числом або складним хешем, тісно пов'язаний з конкретним сеансом користувача. Він включається в запит щоразу, коли користувач взаємодіє з сервером, наприклад, надсилає форму або виконує AJAX-запит.

При використанні техніки AJAX унікальні токени CSRF включаються в HTML-форми, можуть бути частиною прихованого поля форми або міститися в заголовку запиту. Зловмисники не можуть просто надіслати фальшивий запит своїм потенційним жертвам, оскільки їм потрібен доступ до справжнього токену, який є унікальним для кожного користувача.

Коли сервер отримує запит, він перевіряє, чи легітимний токен CSRF і чи належить він конкретному сеансу користувача. Якщо токен недійсний або не відповідає сесії, сервер відхиляє запит, підозрюючи CSRF-атаку. Дійсний токен для конкретного користувача повинен бути доступний зловмисникам, які намагаються здійснити CSRF-атаку. Однак, оскільки токени є унікальними і залежать від сеансу, вони не можуть з легкістю заволодіти відповідним токеном, що ускладнює проведення атаки. Крім того токен можна вбудувати в метатеги, які будуть автоматично додавати токен при запиті до сервера, але викрасти його за допомогою XSS, наприклад, вже не вдасться, оскільки токен не буде вбудований в DOM.

Встановлення атрибуту SameSite в значення "Strict" або "Lax" підвищує рівень безпеки файлів cookie, зменшуючи їх вразливість до атак. Атрибут SameSite має всього два основних значення: "Strict" і "Lax". Коли атрибут

SameSite встановлено в значення "Strict", файл cookie буде включений тільки в запити, відправлені зі сторінки, яку користувач відкрив у поточному вікні браузера. Цей підхід застосовується для того, щоб ускладнити проведення CSRF-атак, оскільки файл cookie не буде додаватися до запитів, що надсилаються з інших джерел.

У режимі Lax файл cookie буде додано до зовнішніх запитів, що надсилаються через зображення або скрипти. Однак, він не буде включений в запити, зроблені при переході користувача з інших джерел. Таким чином досягається баланс між безпекою і зручністю, дозволяючи використовувати файли cookie в певних сценаріях і обмежуючи їх використання в інших. Використовуючи атрибут SameSite, розробники матимуть точний контроль над тим, як файли cookie включаються в різні типи запитів, і можуть вибирати рівень захисту CSRF, який найкраще відповідає конкретним потребам того чи іншого веб-додатку.

Метод Double Submit Cookies заснований на використанні унікального маркера CSRF, який одночасно зберігається в файлі cookie і включається в кожен запит, що надсилається або вбудовується в форму. Мета цього методу - забезпечити механізм подвійного представлення, за допомогою якого сервер може порівнювати значення маркера CSRF, що зберігається в файлі cookie, зі значенням, що міститься в полі форми або додається до заголовка запиту. При реалізації цього методу для кожної сторінки або дії, що виконується, генерується унікальний маркер CSRF.

Токен зберігається в файлі cookie в браузері клієнта і вставляється в форму або додається до інших параметрів запиту. Коли запит надсилається на сервер, значення токена порівнюється зі значенням, що зберігається в файлі cookie. На основі результатів порівняння сервер визначає, чи є запит легальним. Механізм подвійного представлення ускладнює проведення успішної CSRF-атаки, оскільки для її проведення зловмисник повинен не тільки знати значення CSRF-токена, але й успішно взаємодіяти з файлом cookie на стороні потенційної жертви. Це значно

підвищує рівень безпеки веб-додатків і запобігає несанкціонованим операціям з боку автентифікованих користувачів.

Ефективним заходом для захисту від уразливостей CSRF є обов’язкове використання методу POST замість GET для чутливих дій, таких як зміна статусу облікового запису, видалення ресурсів або проведення фінансових операцій. Цей підхід використовує властиві характеристики браузерів, які зазвичай не дозволяють веб-сторінкам автоматично генерувати запити POST за допомогою вбудованих тегів, зокрема тегів Ошибка! Не указано имя файла.

3.3 Запуск автоматизованого тестування на пошук вразливостей web-сервісів

Механізм автоматизованого тестування XSS вразливостей можна реалізувати багатьма способами, наприклад за допомогою PHPUnit і Selenoid, який є зручним докер-контейнером для Selenium. Для початку, варто розглянути ці два інструменти.

PHPUnit - це фреймворк для тестування, який широко використовується і має високу надійність для мови програмування PHP. Він спеціально розроблений для спрощення процесу розробки модульних тестів, які є ключовими елементами в розробці програмного забезпечення. Створений Себастьяном Бергманом, PHPUnit дозволяє розробникам створювати автоматизовані тести, які забезпечують точність окремих блоків або компонентів у кодовій базі PHP. Ці тести необхідні для підтримки цілісності коду, виявлення помилок на ранніх стадіях циклу розробки та запобігання ненавмисному регресу при внесенні змін до коду.

PHPUnit пропонує підтримку класу PHPUnit_Framework_TestCase, який забезпечує структурований підхід до визначення тестових кейсів, а також надає повний набір методів тверджень для перевірки очікуваних результатів. На додаток до базового модульного тестування, PHPUnit також підтримує інші типи тестування, включаючи інтеграційне тестування і функціональне тестування, які сприяють ретельному і всебічному тестуванню. Як невід'ємна частина сучасної PHP-розробки, PHPUnit легко інтегрується з інструментами безперервної

інтеграції, системами контролю версій та іншими компонентами робочого процесу розробки.

Для інсталяції цього фреймворку можна використати пакетний менеджер Composer. Composer - це надійний менеджер залежностей для PHP, який має на меті спростити керування бібліотеками та пакунками у PHPпроектах. Будучи важливим інструментом у сучасній розробці PHP, Composer дозволяє розробникам декларувати бібліотеки, на які покладаються їхні проекти, а також без особливих зусиль керувати встановленням та оновленням цих залежностей. Використовуючи простий конфігураційний файл JSON під назвою "composer.json", розробники можуть вказати вимоги до проекту, такі як версія PHP, розширення та зовнішні пакунки. Composer використовує централізований репозиторій, відомий як Packagist, який містить велику колекцію бібліотек PHP, що можуть бути використані в різних проектах.

Ключова перевага Composer полягає в його здатності ефективно вирішувати і отримувати залежності, забезпечуючи точне встановлення необхідних версій. Крім того, він спрощує автозавантаження, дозволяючи безперешкодно включати класи і функції з встановлених бібліотек. Composer став незамінним інструментом в екосистемі PHP, просуваючи стандартизований та ефективний підхід до управління залежностями, тим самим покращуючи модульність коду, співпрацю та загальний робочий процес розробки. Його широке розповсюдження підкреслює його значення як фундаментального компонента у сучасному середовищі розробки PHP.

Щоб інтегрувати Selenium, спочатку потрібно встановити сервер Selenium і відповідний WebDriver браузера. Можна використовувати окремий сервер Selenium або такі інструменти, як ChromeDriver чи GeckoDriver. Після завантаження та налаштування потрібно переконатися, що сервер Selenium або WebDriver доступні з тестового середовища PHPUnit.

Тестовий файл потрібно розширити класом PHPUnit_Extensions_Selenium2TestCase у тестовому класі PHPUnit, щоб використовувати функції Selenium. Сконфігурувати ініціалізацію і закриття сесії

Selenium, можна визначивши необхідні методи, такі як setUp() і tearDown(). Використання анотацій, такі як @group для категоризації тестів і @dataProvider для параметризованого тестування.

У методі setUp() налаштовуються браузер, можливості браузера, хост і порт для Selenium. Задається базова URL-адреса для веб-додатку і використовуються команди Selenium для взаємодії з браузером під час тестування.

Для написання методів тестування для виконання дій у веб-додатку, таких як натискання кнопок, заповнення форм і перевірки очікуваних результатів використовуються команди Selenium WebDriver, такі як clickOnElement(), type() та твердження для перевірки результатів, тобто асерти.

WebDriver служить мостом між мовою програмування і веббраузерами, дозволяючи розробникам і тестувальникам створювати сценарії, які автоматизують тестування веб-додатків на різних браузерах і платформах. Використовуючи специфічний для браузера драйвер, Selenium WebDriver дозволяє емуляцію взаємодії користувача з веб-елементами, такими як натискання кнопок, заповнення форм, навігація сторінками та перевірка очікуваної поведінки.

Його сумісність з різними браузерами гарантує, що один і той самий набір тестів може бути безперешкодно виконаний, тим самим покращуючи покриття та надійність тестів. Крім того, Selenium WebDriver відіграє важливу роль у підтримці безперервної інтеграції та безперервної доставки, тим самим підвищуючи ефективність та якість процесів розробки програмного забезпечення.

Завдяки своїй універсальності, розширюваності та потужній підтримці спільноти, Selenium WebDriver став незамінним інструментом для професіоналів, які займаються тестуванням веб-додатків та автоматизацією тестування.

Для зручності тестування можна використовувати Selenoid. Selenoid - це високоефективна контейнерна селенова сітка з відкритим вихідним кодом. Її мета - спростити та покращити процес тестування веб-додатків.

На відміну від традиційних Selenium-сіток, Selenoid використовує контейнери Docker, щоб запропонувати легке і масштабоване рішення для запуску браузерних тестів в ізольованих середовищах. Цей унікальний підхід

гарантує, що кожен тест має свій власний виділений браузер і залежності, що допомагає уникнути проблем з розподілом ресурсів і гарантує послідовне і відтворюване середовище тестування.

Selenoid сумісний з різними версіями браузерів і може бути легко інтегрований з популярними тестовими фреймворками, такими як TestNG і JUnit. Його модульна архітектура в поєднанні з можливістю динамічного розподілу ресурсів на основі попиту роблять Selenoid потужним інструментом для організацій, які прагнуть оптимізувати свою інфраструктуру веб-тестування. Крім того, Selenoid надає такі функції, як відеозапис, попередній перегляд у браузері в реальному часі та докладні журнали, які значно полегшують всебічний аналіз результатів тестування.

Щоб створити надійну та ефективну мережу Selenium, установка Selenoid за допомогою AeroCube CM включає в себе ряд комплексних кроків. AeroCube CM, інструмент управління конфігурацією, спрощує розгортання та управління Selenoid, реалізацією Selenium grid з відкритим вихідним кодом.

Перед початком процесу інсталяції важливо переконатися, що AeroCube CM належним чином встановлений у системі. Якщо це так, можна переходити до створення конфігураційного файлу, в якому будуть вказані бажані налаштування для Selenoid, такі як версії браузерів, роздільна здатність екрану і мережеві параметри. Після цього, варто перейти до розгортання проєкту за допомогою інтуїтивно зрозумілого інтерфейсу AeroCube CM, інтегрувавши Selenoid в існуючу інфраструктуру. Крім того, слід використовувати можливості моніторингу та масштабування AeroCube CM для ефективного управління ресурсами і адаптації до різних вимог тестування. Після успішного завершення інсталяції рекомендовано провести ретельне тестування, щоб перевірити функціональність і стабільність мережі Selenium на базі Selenoid. Ця комплексна установка, що поєднує сильні сторони AeroCube CM і Selenoid, створює відмовостійке і масштабоване середовище для тестування веб-додатків, підвищуючи загальну ефективність тестування і забезпечуючи безперебійний досвід тестування.

Додатковою корисною опцією є підтримка VNC імеджів браузерів в селеноїді. Щоб увімкнути підтримку VNC, необхідно включити необхідні параметри VNC у конфігурацію, а також завантажити необхідні образи VNC для браузера.

Перевагу надають образам Chrome і Firefox, які попередньо налаштовані з підтримкою VNC. Потрібно використати команди Docker, щоб витягнути потрібні образи з Docker Hub на локальний комп'ютер і переконатися, що версії браузерів відповідають версіям, зазначеним у конфігурації Selenoid.

Після успішного налаштування тестового середовища, можна переходити до написання автоматизовних тестів для перевірки XSS. Для тестування XSS-уразливостей можна створити PHPUnit-тест за допомогою Selenium, розробивши тестовий сценарій, який має на меті впровадити шкідливий вміст у веб-додаток. Важливо відзначити, що проведення таких тестів повинно здійснюватися відповідально і виключно на системах, де було надано явний дозвіл на тестування.

Припустимо, що є поле форми, яке може бути вразливим до XSS. Тест вводить ретельно розроблене корисне навантаження, щоб перевірити, чи ефективно додаток обробляє та запобігає XSS-атакам. Сам тест виглядає наступним чином:

Для початку потрібно визначити простір імен і включають необхідні класи з PHPUnit і бібліотеки Facebook WebDriver.

```
use PHPUnit\Framework\TestCase;
use Facebook\WebDriver\Remote\RemoteWebDriver;
use Facebook\WebDriver\WebDriverBy;
```

Після цього оголосити клас з іменем XssTest. Цей клас розширює клас TestCase, що надається PHPUnit. Його призначенням є визначення та виконання тестів відповідно до вимог PHPUnit.

```
class XssTest extends TestCase
```

Оголосити змінну в класі у якій будемо зберігати об’єкт веб-драйвера.

```
private $webDriver;
```

Описати метод setUp, що викликається перед кожним тестом. Він ініціалізує WebDriver, створюючи віддалене з'єднання з сервером Selenium, що працює за адресою http://localhost:4444/wd/hub, за допомогою браузера Chrome.

```
protected function setUp(): void
   {
      $this->webDriver
RemoteWebDriver::create('http://localhost:4444/wd/hub', ['platform' => 'LINUX',
 'browserName' => 'chrome']);
       }
```

І описати сам тесткейс.

```
 public function testXssVulnerability()
{
// Завантажити веб-сторінку
$this->webDriver->get('http://your-app-url');
 //Визначити форму використовуючи її атрибут name
$textField = $this->webDriver->findElement(WebDriverBy::name('input_field_name'));
// Ініціалізуємо скрипт з вразливістю
$xssPayload = '';
// Вводимо підготовлений скрипт в форму
$textField->sendKeys($xssPayload);
// Відправляємо ворму натисканням кнопки submit
$this->webDriver->findElement(WebDriverBy::name('submit_button'))- >click();
// Очікуємо 2 секунди на відповідь сторінка
```

```
sleep(2);
// Перевіряємо чи скрипт виконався
$this->assertTrue($this->webDriver->switchTo()->alert()->getText() === 'XSS');
  }
```

Після цього описуємо метод tearDown що завершує сесію веб-драйвера. protected function tearDown(): void { $this->webDriver->quit(); }

Висновки до третього розділу

В розділі 3 було запропоновано створення безпекових рекомендацій у вигляді готових рішень для захисту від WEB-атак класичним пристроєм є Web Application Firewall, який застосовує набір правил захисту до протоколів високого (прикладного) рівня HTTP/HTTPS, FTP/FTPS. Класичне розміщення WAF в мережі – в режимі зворотного проксісервера перед захищеними веб-серверами. Але цього не достатньо, тому в роботі пропонується комплексне рішення, яке включає WAF, IPS та FIREWALL, тобто захист на всіх рівнях моделі OSI.

Також було виключено додаткові засоби боротьби від атак, які будуть спроможні захистити Web ресурси організацій, а для більш вдалого моніторингу актуальних загроз, створення автоматичнох системи тестування.

ВИСНОВКИ

В кваліфакіційній роботі отримано наступні теоретичні та наукові результати:

– досліджено, що web-сервіси є дуже вразливими для шахраїв, та мають необхідність у своєму захисті, в останні роки злам web-сервісів організацій став буденним явищем, характерним як для розвинутих країн, так і для країн, що розвиваються. Основним об'єктом бажань зловмисників, природно, є інформація, оброблювана в різних організаціях.

– проаналізовано, що WAF та IPS можуть стати ефективним інструментом для захисту веб сервісів будь-якої організації від широкого спектру кіберзагроз. Вони можуть допомогти запобігти зараженню зловмисним програмним забезпеченням, фішинговим атакам та іншим зловмисним діям, перш ніж він зможе потрапити до веб сервісів організацій.

– зазначено, що згідно з проведенною роботою, рекомендується застосовувати WAF та IPS як ефективний засіб для захисту веб сервісів організацій від широкого спектру кіберзагроз, включаючи зараження зловмисним програмним забезпеченням, фішингових атак та іншої шкідливої діяльності.

ПЕРЕЛІК ПОСИЛАНЬ

1. Guru99 [Електронний ресурс]– Режимдоступу: https://www.guru99.com/uk/web-service-architecture.html
2. The Web Application Security Consortium (WASC) – Articles [Електронний ресурс]–Режимдоступу: http://www.webappsec.org/projects/articles/
3. The Web Application Security Consortium (WASC) [Електронний ресурс]–Режимдоступу: http://www.webappsec.org/
4. White Hat Security testing. Improper Input Handling. [Електронний ресурс]–Режимдоступу: https://www.whitehatsec.com/glossary/content/improperinput-handling
5. WhiteHat Security's Approach to Detecting Cross-Site Request Forgery(CSRF) [Електронний ресурс]–Режимдоступу: https://www.whitehatsec.com/
6. Security from CSRF. [Електронний ресурс]–Режимдоступу: https://www.owasp.org/index.php/CrossSite_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
7. OWASP - Cross Site Scripting (XSS) [Електронний ресурс]– Режимдоступу: https://owasp.org/www-community/attacks/xss/
8. XSS Secured WebApplications by using innerHTML Mutations [Електронний ресурс]–Режимдоступу: https://security.stackexchange.com/questions/46836/what-is-mutation-xssmxss
9. OWASP: Vulnerability Scanning Tools. [Електронний ресурс]– Режимдоступу: https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools
10. Дослідження вразливостей Web-сайтів та методів їх усунення [Електронний ресурс]–Режимдоступу: http://phone.kpi.ua/wpcontent/uploads/2014/06/4.pdf
11. Top 25 Most Dangerous Software Errors. [Електронний ресурс]– Режимдоступу: : https://www.sans.org/top25-softwareerrors

12. Загрози безпеці і способи захисту інтернет ресурсів. [Електронний ресурс]–Режимдоступу: https://adgrafics.net/certificate_security.html