МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА

ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В ЕКОНОМІЦІ

Кафедра системного аналізу та кібербезпеки

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КІБЕРБЕЗПЕКА»
Галузь знань 12 «Інформаційні технології»
Спеціальність 125 «Кібербезпека»
Форма навчання: очна (денна)

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему **«Розробка та впровадження комплексних методів захисту веб-застосунків від кібератак»**

здобувача Марциновського Семена Михайловича

Науковий керівник: д.т.н., професор Толюпа Сергій Васильович

_____
(підпис)

Робота допущена до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти (ЕК)

Завідувач кафедри: д.ф.-м.н., професор Джалладова І.А.

_____
(підпис)

Київ 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА

ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В ЕКОНОМІЦІ

Кафедра системного аналізу та кібербезпеки

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КІБЕРБЕЗПЕКА»
Галузь знань 12 «Інформаційні технології»
Спеціальність 125 «Кібербезпека»
Форма навчання: очна (денна)

ПОГОДЖЕНО:	**ЗАТВЕРДЖУЮ:**
Керівник проектної групи (гарант) освітньо-професійної програми «Кібербезпека» к.ф.-м.н, доцент Г.В. Мамонова	Завідувач кафедри системного аналізу та кібербезпеки д.ф.-м.н., проф. І.А. Джалладова
_____	_____
_____2024 р.	_____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти Марциновського Семена Михайловича

денної форми навчання

на підготовку кваліфікаційної бакалаврської роботи

на тему **«Розробка та впровадження комплексних методів захисту веб-застосунків від кібератак»**

Тему затверджено наказом ректора Університету від 30.04.2024р. № 726-ст

Кваліфікаційна бакалаврська робота виконується на матеріалах

_____

_____

ПЛАН КВАЛІФІКАЦІЙНОЇ БАКАЛАВРСЬКОЇ РОБОТИ

Розділ 1	Основи захисту веб застосунків
Розділ 2	Аналіз існуючих рішень у сфері кібербезпеки
Розділ 3	Розробка надійного веб-застосунку
Об'єкт дослідження:	Функціональні особливості та вразливі місця веб-сайтів та веб-застосунків
Предмет дослідження: спеціалізоване програмне забезпечення для виявлення кібератак та моніторингу підозрілої активності	
Мета кваліфікаційної бакалаврської роботи: захист інформації веб-застосунків від кібератак за допомогою спеціалізованого програмного забезпечення	
Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети: 1. Дослідити концепцію та вимоги до безпеки веб-додатків. 2. Проаналізувати принципи та технології кібербезпеки у вебі. 3. Проаналізувати минулі випадки атак на відомі веб-додатки. 4. Розробити повноцінний дизайн системи для додатку веб-гаманця 5. Перелічити та вибрати найкращі інструменти для забезпечення максимального захисту веб-гаманця	
У розділі 1	Було визначено основні види кібератак, включаючи SQL-ін'єкції, Cross-Site Scripting (XSS), Denial of Service (DoS), атаки типу Man-in-the-Middle (MitM) та Cross-Site Request Forgery (CSRF). Розглянуто моделі безпеки веб-застосунків, такі як модель загроз і оцінка ризиків, а також регулятивні вимоги та стандарти безпеки, включаючи GDPR, PCI DSS та ISO/IEC 27001.
У розділі 2	Було розглянуто популярні інструменти для захисту веб-застосунків, такі як WAF, RASP, та IDS/IPS системи, а також проаналізовано ефективність різних методів захисту, включаючи порівняльний аналіз WAF та RASP, оцінку проникнення та комплексні рішення з кібербезпеки. Окрім цього, було розглянуто найвідоміші кібератаки у вебі, зокрема SQLi атаку на Sony Pictures та XSS атаку на Твіттер.
У розділі 3	Було з'ясовано проблему та встановлено обсяг роботи, розроблено поверхневий дизайн системи, включаючи дизайн АПІ, рішення для сегментування в пам'яті та Event Sourcing. Здійснено поглиблення у дизайн системи та вибір технологій і підходів з акцентом на безпеку.

Завдання підготував
науковий керівник

_____ д.т.н., професор Толюпа С.В.

Завдання одержав
здобувач

_____ Марциновський С.М.

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Розробка та впровадження комплексних методів захисту веб-застосунків від кібератак» складається зі списку скорочень, вступу, основної частини, що містить 3 розділи, висновку й списку літератури. Загальний обсяг роботи - 75 сторінки. Робота містить 28 рисунків та 3 таблиці. Список використаних джерел та літератури включає 20 джерел.

Об'єкт дослідження - функціональні особливості та вразливі місця веб-сайтів та веб-застосунків.

Предмет дослідження - спеціалізоване програмне забезпечення для виявлення кібератак та моніторингу підозрілої активності.

Мета роботи - захист інформації веб-застосунків від кібератак за допомогою спеціалізованого програмного забезпечення.

Метод дослідження - аналіз механізмів виявлення кібератак за допомогою спеціалізованого програмного забезпечення.

В роботі проведено аналіз функціональних особливостей існуючих програмних продуктів та за рядом критеріїв визначено варіант для реалізації захисту веб-застосунків.

Практичне значення роботи полягає у визначенні ефективного за рядом критеріїв програмного продукту в сфері захисту веб-застосунків серед відомих виробників, виявленні переваг та недоліків існуючих програмних засобів, окресленні шляхів подальшого покращення засобів захисту веб-застосунків.

Результати здійснених у дипломній роботі досліджень можуть бути використані спеціалістами із захисту інформації з метою підвищення безпеки веб-ресурсів організації та при подальшому проведенні науково-дослідницьких робіт в даній області.

Напрямки подальших досліджень: визначення та пошук шляхів для боротьби з веб кіберзлочинністю

Ключові слова: веб-сайт, веб-застосунок, SSL, веб-клієнт, веб-сервер, компоненти Web, API, HTML вразливість , Web Application Firewall, кібератака.

ВІДГУК

на кваліфікаційну бакалаврську роботу
студента: Марциновського Семена Михайловича гр. ІК - 401
на тему: **«Розробка та впровадження комплексних методів захисту веб-застосунків від кібератак»**

Актуальність теми: кваліфікаційна бакалаврська робота Марциновського Семена присвячена **розробці та впровадженню комплексних методів захисту веб-застосунків від кібератак**.

Цифровий простір ще ніколи не був настільки небезпечним, як сьогодні. Кількість та складність кібератак росте з року в рік, тоді як організації публічного, державного та приватного секторів дуже часто не мають необхідних ресурсів та досвіду для налагодження комплексного захисту. Однією з найбільших проблем залишаються атаки на веб- та мобільні додатки, що можуть скомпрометувати дані в системі, порушити транзакції, чи навіть повність паралізувати бізнес. Більш сучасні системи – такі як IPS / IDS (система запобігання вторгнень / система виявлення вторгнень) – здатні аналізувати вміст мережевих пакетів і порівнювати його з сигнатурами відомих атак. Крім того, ці системи виявляють і блокують відхилення в протоколах прикладного рівня. Однак сьогодні понад 80% атак експлуатують уразливості додатків, а не мережевий архітектури. До того ж, сьогодні існує величезна кількість веб-додатків (кожне з яких потенційно може мати які-небудь уразливими), тобто, загальне число вразливостей набагато більше, ніж кількість сигнатур в базах сучасних IPS – систем. За оцінками експертів, саме проникнення через веб-додатки останнім часом стають основним вектором атак на корпоративні мережі. Таким чином тема є актуальною і потребує детального дослідження.

Повнота розкриття теми: в даній роботі повністю виконані поставлені завдання, її обсяг та зміст відповідає задачам на бакалаврську кваліфікаційну роботу.

Теоретичний рівень: проведене дослідження свідчить про вміння автора опрацьовувати та аналізувати іншомовні видання, нормативні документи, технічну документацію, проводити аналіз, використовувати метод індукції та узагальнювати матеріал.

Практичне значення роботи полягає у визначенні ефективного за рядом критеріїв програмного продукту в сфері захисту веб-застосунків серед відомих виробників, виявленні переваг та недоліків існуючих програмних засобів, окресленні шляхів подальшого покращення засобів захисту веб-застосунків.

Самостійність виконання роботи: студент показав здатність самостійно працювати з науково-технічною літературою, визначати вектор дослідження, знаходити та обробляти інформацію, має достатні теоретичні знання. Робота відповідає затвердженій темі, а всі етапи, затверджені календарним планом, були виконані у зазначений термін.

Якість оформлення, загальна та спеціальна грамотність: пояснювальна записка кваліфікаційної бакалаврської роботи написана якісно та в доступній для ознайомлення формі.

Переваги та недоліки роботи: *теоретична значущість* дослідження полягає у систематизації та узагальненні сучасних знань про методи захисту веб-застосунків, а також у розробці новітніх підходів до забезпечення кібербезпеки; м*етодична значущість* полягає у впровадженні ефективних практичних рішень, які можуть бути використані розробниками для створення захищених веб-застосунків.

Загальна оцінка роботи та висновок щодо рекомендації до захисту в ЕК: вважаю, що кваліфікаційна робота виконана на достатньо професійному рівні і студент Марциновський Семен підтвердив свою підготовленість до самостійної роботи в галузі інформаційної безпеки, і заслуговує на оцінку «добре/82» та присвоєння йому ступеня бакалавра зі спеціальності «Кібербезпека».

Науковий керівник
професор кафедри
комп'ютерної математики та
інформаційної безпеки ННІ
«Інститут інформаційних
технологій в економіці»
д.т.н., професор _____ Сергій ТОЛЮПА
(підпис) *(ініціали, прізвище)*

«10» червня 2024 р.

РЕЦЕНЗІЯ

на кваліфікаційну бакалаврську роботу

студента: Марциновського Семена Михайловича гр. ІК - 401
на тему: **«Розробка та впровадження комплексних методів захисту веб-застосунків від кібератак»**

Актуальність і перспективність роботи - тематика роботи є надзвичайно актуальною в умовах сучасного розвитку цифрових технологій. Кількість веб-застосунків зростає щоденно, як і кількість загроз, що на них спрямовані. Зважаючи на це, розробка надійних методів захисту веб-застосунків стає пріоритетним завданням для фахівців з кібербезпеки. Перспективність тематики полягає у можливості створення інноваційних рішень, що забезпечать високий рівень захисту та сприятимуть розвитку безпечного інформаційного простору. Виходячи з цього можна стверджувати, що кваліфікаційна бакалаврська робота є актуальною та своєчасною.

Науково новизна роботи полягає у розробці та впровадженню комплексних методів захисту веб-застосунків від кібератак. В роботі розглянуто основи криптографічного захисту даних, включаючи симетричне та асиметричне шифрування, хешування та цифрові підписи, а також роль SSL/TLS сертифікатів у забезпеченні безпеки веб-застосунків. Було проаналізовано різні моделі безпеки веб-застосунків, такі як модель загроз та оцінка ризиків, що дозволило зрозуміти, як ідентифікувати та мінімізувати потенційні загрози. Оглянули регулятивні вимоги та стандарти безпеки, такі як GDPR, PCI DSS та ISO/IEC 27001, які встановлюють вимоги до захисту даних та забезпечують правові рамки для їх реалізації.

Якість проведеного аналізу. Здобувач показав достатньо високий рівень володіння положеннями обраної теми, здатність формувати власну точку зору на основі аналізу результатів дослідження різних науковців цієї області.

Уміння користуватися літературними джерелами. В ході роботи її автором була опрацьована достатня кількість літературних джерел. На високому теоретичному та методологічному рівнях розглянуто питання розробці та впровадженню комплексних методів захисту веб-застосунків від кібератак.

Практична цінність висновків і рекомендацій полягає у визначенні ефективного за рядом критеріїв програмного продукту в сфері захисту веб-застосунків серед відомих виробників, виявленні переваг та недоліків існуючих програмних засобів, окресленні шляхів подальшого покращення засобів захисту веб-застосунків.

Переваги та недоліки. Беззаперечною перевагою роботи є її практична цінність. Проте, нажаль, виявлені незначні недоліки кваліфікаційної бакалаврської роботи, що полягають в наявності помилок орфографічного та синтаксичного плану. Однак, це в цілому, не впливають на якість дослідження даної теми та її результативність.

Загальний висновок і оцінка роботи. Аналіз запропонованої роботи свідчить про бачення автором цілісної картини проблем «Розробка та впровадження комплексних методів захисту веб-застосунків від кібератак», це дозволяє зробити висновок про те, що кваліфікаційна бакалаврська робота студента Марциновського Семена відповідає всім вимогам, що висуваються до кваліфікаційних бакалаврських робіт. Робота виконана на високому науковому та теоретичному рівнях і заслуговує оцінки «добре», а автору роботи може бути присвоєно освітній рівень «бакалавр» за спеціальністю Кібербезпека.

РЕЦЕНЗЕНТ
професор кафедри інформаційних систем та технологій
факультету інформаційних технологій КНУ імені Шевченка

д.т.н., професор Володимир ДРУЖИНІН

«10» червня 2024р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

XSS (Cross Site Scripting) - міжсайтове виконання сценаріїв;

MitM (Man-in-the-Middle) - Атака "людина посередині"

DoS (Denial of Service) - Відмова в обслуговуванні

HTML (Hyper Text Markup Language) - мова розмітки гіпертекстових документів;

IP (Internet Protocol) - інтернет-протокол;

IPS (Intrusion Prevention Systems) - система виявлення вторгнень;

SSL (Secure Sockets Layer) - рівень захищених сокетів;

SQL (Structed Query Language) - мова структурованих запитів;

WAF (Web Application Firewall) - брандмауер web-додатків;

TLS (Transport Layer Security) - протокол захисту транспортного рівня

DOM (Document Object Model) - об’єктна модель документа

CSRF(Cross-Site Request Forgery) - Підробка міжсайтового запиту

AES (Advanced Encryption Standard) - Розширений стандарт шифрування

RC (Rivest Cipher) - Шифр Рівеста

DES (Data Encryption Standard) - Стандарт шифрування даних

HTTP (Hypertext Transfer Protocol) - Протокол передачі гіпертексту

HTTPS (Hypertext Transfer Protocol Secure) - Захищений протокол передачі гіпертексту

RASP (Runtime Application Self-Protection) - Самозахист програм на рівні виконання

API (Application Programming Interface) - Інтерфейс програмування застосунків

ВСТУП

Оцінка сучасного стану проблеми - У сучасному світі стрімкого розвитку інформаційних технологій та інтернет-комунікацій, проблема забезпечення безпеки веб-застосунків набуває особливого значення. Аналіз вітчизняної та зарубіжної наукової літератури свідчить про постійне зростання кількості кібератак та їх складності. За даними досліджень, такі типи атак, як SQL ін'єкції, Cross-Site Scripting (XSS), Denial of Service (DoS), Man-in-the-Middle (MitM) атаки та Cross-Site Request Forgery (CSRF) є найбільш поширеними і загрозливими для веб-застосунків.

Провідні вчені та фахівці у сфері кібербезпеки, такі як Брюс Шнайєр, Джонатан Зіттрейн та Брайан Кребс, активно досліджують та розробляють методи захисту від зазначених атак. Незважаючи на численні досягнення, залишається низка нерозв'язаних проблем, серед яких забезпечення ефективного захисту даних у реальному часі та розробка новітніх криптографічних методів.

Актуальність і перспективність даної роботи - тематика роботи є надзвичайно актуальною в умовах сучасного розвитку цифрових технологій. Кількість веб-застосунків зростає щоденно, як і кількість загроз, що на них спрямовані. Зважаючи на це, розробка надійних методів захисту веб-застосунків стає пріоритетним завданням для фахівців з кібербезпеки. Перспективність тематики полягає у можливості створення інноваційних рішень, що забезпечать високий рівень захисту та сприятимуть розвитку безпечного інформаційного простору.

Об’єкт дослідження - функціональні особливості та вразливі місця веб-сайтів та веб-застосунків.

Предмет дослідження - спеціалізоване програмне забезпечення для виявлення кібератак та моніторингу підозрілої активності.

Мета роботи - захист інформації веб-застосунків від кібератак за допомогою спеціалізованого програмного забезпечення.

Метод дослідження - аналіз механізмів виявлення кібератак за допомогою спеціалізованого програмного забезпечення.

Теоретична значущість дослідження полягає у систематизації та узагальненні сучасних знань про методи захисту веб-застосунків, а також у розробці новітніх підходів до забезпечення кібербезпеки.

Методична значущість полягає у впровадженні ефективних практичних рішень, які можуть бути використані розробниками для створення захищених веб-застосунків.

ОСНОВИ ЗАХИСТУ ВЕБ ЗАСТОСУНКІВ

1.1. Визначення кібератаки та її види

SQL ін'єкції

SQL використовується для отримання та обробки даних у базі даних. SQL-запит - це спосіб вставляти, змінювати, витягувати та видаляти дані в базі даних. SQL-ін’єкція - атака, під час якої зловмисник втручається у виконання запитів.

Атакуючий може виконувати цю атаку з різними намірами, такими як:

1. Вилучення даних: Атакуючі можуть витягувати конфіденційну інформацію, збережену в базі даних. Наприклад, якщо атакуючий отримує доступ до адміністративної бази даних.
2. Екстракція даних - Чутливі дані будуть захоплені атакуючим. Припустимо, якщо базу даних адміністратора зламано, вся база даних стає вразливою.
3. Доступ до даних - Вони намагаються зламати привілеї і отримати доступ до всієї бази даних, а також намагаються маніпулювати даними.
4. Визначення відбитка бази даних - У цій атаці версія бази даних та її тип будуть визначені атакуючим. Ця атака допомагає їм спробувати різні типи запитів у різних додатках.
5. Знаходження ін'єкційних параметрів – за допомогою деяких автоматичних інструментів будуть знайдені вразливі параметри для атаки.
6. Обхід аутентифікації – механізми аутентифікації додатку будуть обійдені для входу в базу даних.

7. Ідентифікація схеми бази даних - з назви таблиці бази даних, типу даних кожного поля, назви стовпця тощо будуть отримані дані для успішного збору інформації.

8. Виконання відмови в обслуговуванні – Видалення таблиці та вимкнення системи входить до цієї категорії. Атакуючий намагається проникнути в систему, щоб виконати певну інструкцію в базі даних.

Види sql-ін'єкцій

1. Атака sql-ін'єкції на основі тавтологій: у логічній математиці тавтологія - це формули, які можливі в усіх можливих випадках. У атаці на основі тавтології код вводиться за допомогою умовного оператора OR, щоб запит завжди оцінювався як істина. Основна мета цієї форми впровадження sql полягає в тому, щоб визначити параметри, які можна ввести, обійти автентифікацію та витягти дані

2. Sql injection union attack: коли програма вразлива до sql-ін’єкцій і результати запиту повертаються у відповідях програми, union. Ключове слово можна використовувати для отримання даних з інших таблиць у базі даних. Це призводить до sql-атаки union. Цей тип атаки використовує оператор union (u) під час вставки запиту sql. Два sql-запити об’єднані оператором union. Перший оператор - це звичайний запит, після якого до шкідливого запиту додається оператор об’єднання. Отже, він використовується для обходу механізму запобігання та виявлення системи.

Cross-Site Scripting (XSS)

Міжсайтовий скриптинг - це тип атаки, який використовується для отримання доступу до браузера жертви за допомогою вразливостей у веб-додатку, отримання доступу до приватної та конфіденційної інформації користувача. XSS – це атака з ін’єкцією коду, яка відбувається на стороні клієнта. Зловмисник атакує веб-браузер жертви, запускаючи шкідливий сценарій у веб-додатку.

Якщо веб-програма використовує недезінфікованого користувача та відображає те саме, що виводить, цю програму можна вважати вразливою до

атаки XSS. XSS-атаки найчастіше зустрічаються в JavaScript. Однак це можливо в CSS, VBScript, ActiveX тощо.

ПРИКЛАД. Загальнодоступний блог або розділ коментарів форуму приймає дані користувачів, зберігає їх у їхній базі даних і відображає той самий вміст іншим користувачам, які відвідують блог чи форум. Ці типи програм є ідеальною мішенню для зловмисників, які можуть вставити шкідливий сценарій у поле введення та надіслати його, щоб сценарій було збережено в базі даних. Щоразу, коли користувач відвідує блог або форум, сценарій виконується в браузері цього користувача як частина веб-сторінки, надаючи зловмиснику доступ до конфіденційних даних користувача.

Веб-сторінки, які використовують SSL для шифрування обміну даними, також можуть бути цілями для XSS, оскільки шкідливий сценарій виконується в контексті DOM веб-додатку.

Типи XSS атак:

Цю форму XSS-уразливості іноді називають **постійною** XSS. Це пов’язано з тим, що шкідливий скрипт все ще присутній на сервері після завершення атаки. Під час цього типу атаки зловмисник впроваджує код, який був зловмисно написаний на сервері таким чином, що його неможливо видалити. Як показано на рисунку 1.1, сценарій, який я використовував для ілюстрації збереженої атаки XSS, вводив тег сценарію безпосередньо в об’єктну модель документа (DOM), а потім гіпотетично виконував шкідливий сценарій за допомогою JavaScript. Однак, незважаючи на те, що це найпопулярніший метод використання XSS, це також найпоширеніший підхід, який нейтралізують досвідчені фахівці з безпеки та розробники програмного забезпечення, які піклуються про безпеку. Користувач завантажує зловмисний сценарій XSS до бази даних, який запитують і переглядають інші користувачі, що призводить до виконання сценарію в їхніх системах, як описано на рисунку 1.1.

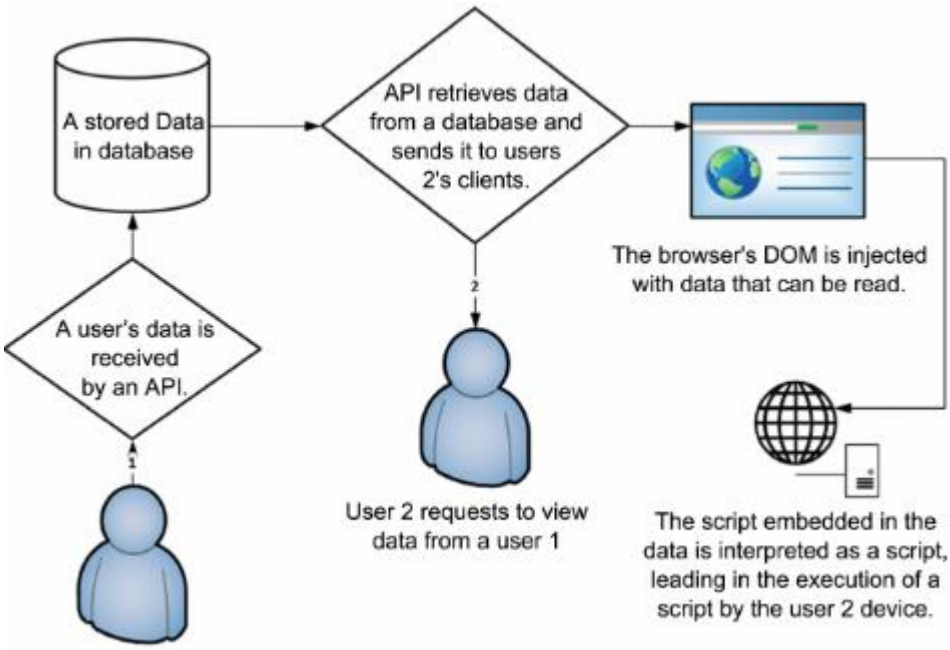


Рисунок 0.1 - XSS атака

Відображена(reflected) атака XSS, яка також відома як непостійна атака, полягає в тому, щ о зловмисник генерує URL-адресу, яка вставляє довільні сценарії в цільову веб-програму. Більшість публікацій і наукових ресурсів вводять відображені XSS перед переходом до збережених концепцій XSS. Я вважаю, що відображені XSS-атаки часто важче для нових недосвідчених програмістів виявити та використати, ніж збережені XSS-атаки. Збережена XSS-атака відносно проста для розуміння з точки зору розробника. Клієнт надає ресурс серверу, що зазвичай робиться через протокол HTTP. Сервер вставляє запитуваний ресурс у базу даних після отримання його від клієнта. Потім шкідливий сценарій буде ненавмисно виконано в Інтернет-браузері клієнта, якщо інші клієнти пізніше отримають доступ до цього ресурсу, як показано на рисунок 1.1. З іншого боку, відображені XSS-атаки працюють як збережені XSS-атаки, але не потребують бази даних або сервера. Жоден сервер не бере участі у відображеній атаці XSS, оскільки код клієнта впливає безпосередньо в браузері, як показано на рисунку 1.2. Веб-додатки можуть бути вразливими до цього типу атак (див. рисунок 1.2) через дії, вжиті користувачем який виконує незбережений (взаємозв’язаний) сценарій на комп’ютері користувача.

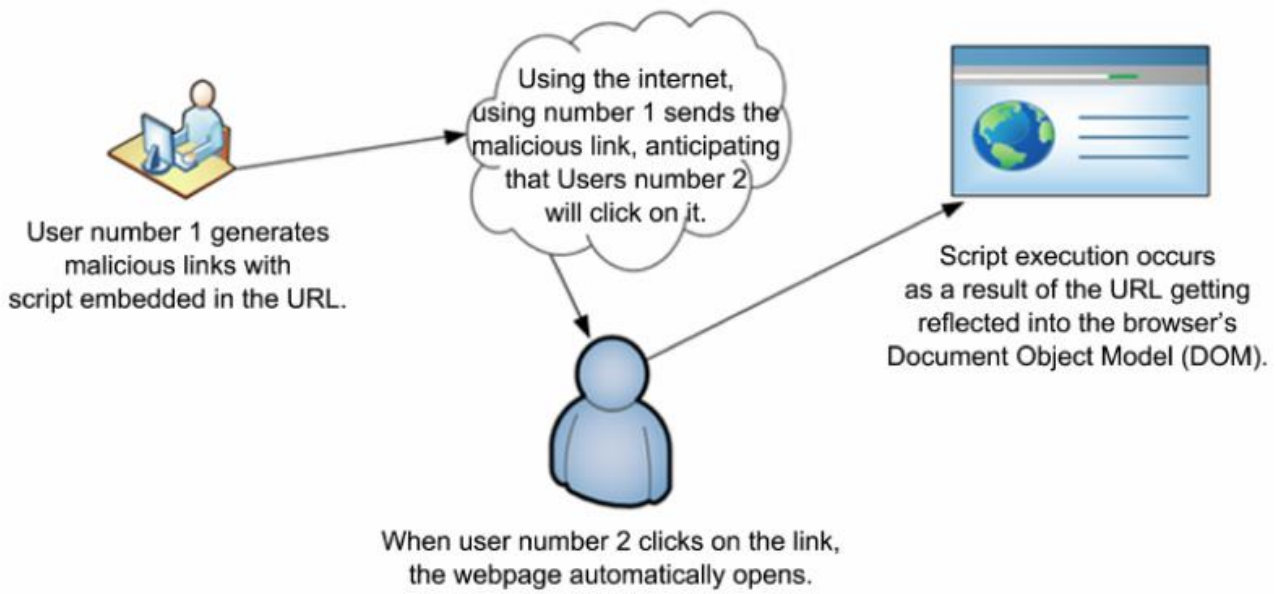


Рисунок 0.2 - Як працює XSS

XSS-атака на **основі DOM**, очевидно, є атакою на стороні клієнта. Тип XSS-атаки на основі DOM зображено на рисунку 1.3 як третя важлива класифікація XSS-атак. Реалізація DOM у різних браузерах може зробити деякі браузери вразливими, а інші – ні. Порівняно з типовими відображеними або збереженими XSS-атаками (рис. 1.1 і рис. 1.2), ці XSS-атаки вимагають глибокого розуміння DOM і JavaScript веб-переглядача, щоб їх виявити та використати. XSS-атаки на основі DOM принципово відрізняються від інших типів XSS тим, що вони жодним чином не вимагають зв'язку з сервером. Відповідно до домовленості, джерелом зазвичай є об'єкт DOM, який може зберігати текст, а приймачем, як правило, є API DOM, який може виконувати сценарій, який зберігається як текст. І «джерело», і «приймач» повинні бути присутніми в DOM браузера, щоб DOM XSS працював, оскільки сервер не задіяний. У більшості випадків приймачем є DOM API, який може запускати сценарій, збережений у джерелі як текст. Майже неможливо виявити DOM XSS за допомогою інструментів статичного аналізу чи інших популярних сканерів, оскільки він ніколи не торкається сервера

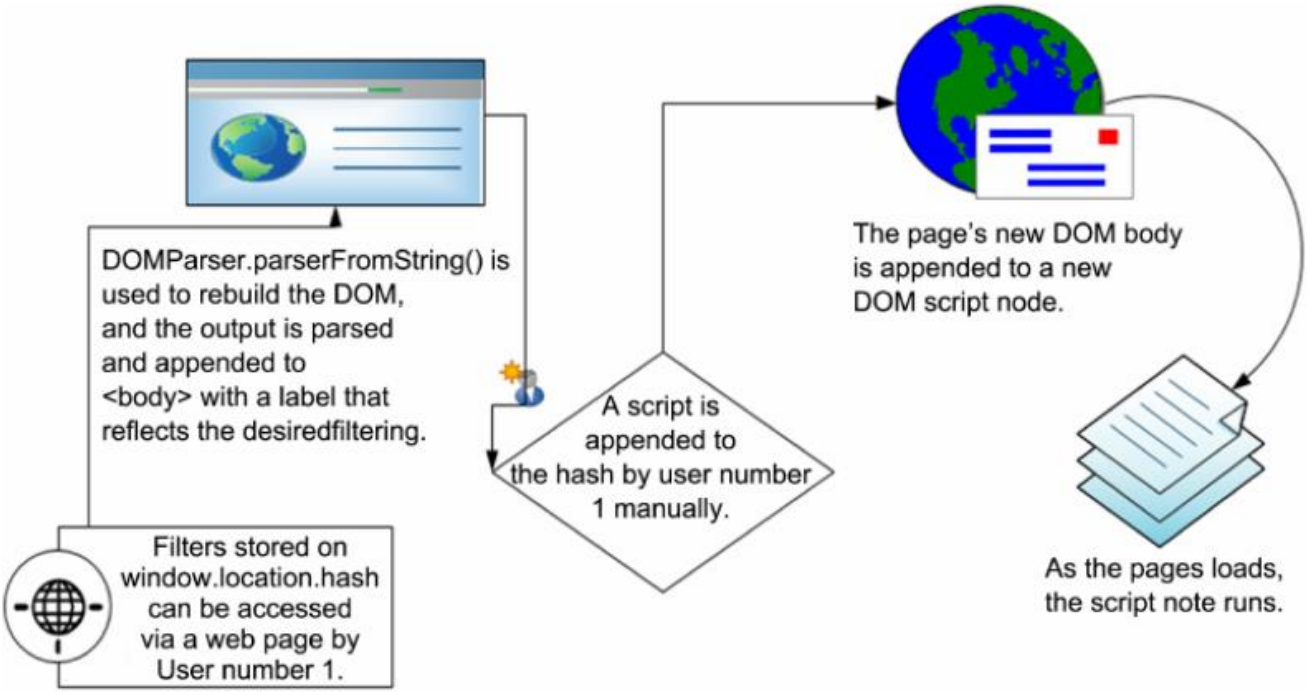


Рисунок 0.3 - DOM XSS

Доктор Маріо Хайдеріх оприлюднив шість нових підкласів атак mXSS у своїй публікації. У атаці mXSS DOM можна повністю уникнути за допомогою InnerHTML, який дозволяє автоматично змінювати вміст HTML. mXSS іноді називають мутованим XSS або XSS на основі мутацій. Це пов’язано з тим, що його важко передбачити і передбачає рекурсію. Коли сценарій HTML завантажується в об’єктну модель документа браузера, дані змінюються, що спричиняє помилку. Однак вміст, завантажений у DOM браузера, змінюється, щоб перевірити, що він без помилок і не містить неправильної розмітки. Це досягається за допомогою атрибута element.innerHTML. Фундаментальним недоліком цієї форми XSS-атаки є її здатність обходити захист на стороні сервера та фільтри на стороні клієнта. На рисунку 1.4 зображено потенційний сценарій XSS-атак на основі мутацій.

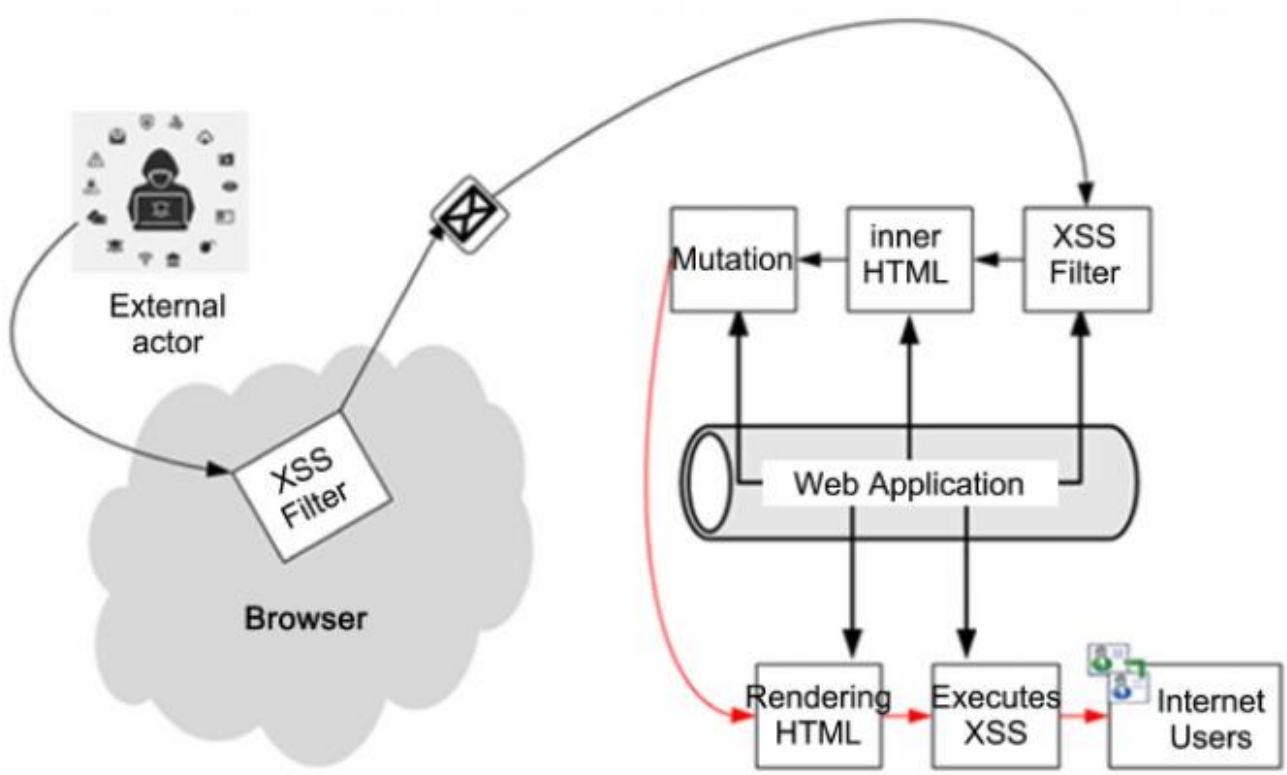


Рисунок 0.4 - Мутаційна XSS

Коли зовнішній актор вводить щось, що здається безпечним, як показано на рисунку 1.4, браузер переписує та змінює це під час обробки HTML, що призводить до мутованої атаки XSS. Це неймовірно ускладнює пошук та усунення помилок у логіці програми. Незважаючи на новизну та поширене неправильне тлумачення, атаки mXSS використовувалися для обходу найскладніших доступних фільтрів XSS. mXSS використовувався для обходу таких рішень, як DOMPurify, OWASP AntiSamy та Google Caja, і велика кількість популярних веб-програм (особливо поштових клієнтів) виявилися вразливими. В основі роботи mXSS використовуються безпечні для фільтрів корисні навантаження, які після фільтрації перетворюються на незахищені корисні навантаження. Усі основні браузери вразливі до атак mXSS. Розробники повинні розуміти, як браузери обробляють оптимізацію та умовні вирази під час відтворення вузлів DOM.

Denial of Service (DoS)

Атаки на відмову в обслуговуванні (DoS) стали основною загрозою для сучасних комп’ютерних мереж. Ранні атаки DoS були технічними іграми серед

підпільних зловмисників. Наприклад, зловмисник може захотіти отримати контроль над IRC-каналом, виконавши DoS-атаки проти власника каналу. Зловмисники могли отримати визнання в підпільній спільноті через видалення популярних веб-сайтів. Оскільки прості у використанні інструменти DoS, такі як Trinoo (Dittrich 1999), можна легко завантажити з Інтернету, звичайні користувачі комп'ютерів також можуть стати DoS-атаками. Іноді вони координовано висловлювали свої погляди, запускаючи DoS-атаки проти організацій, з політикою яких вони не погоджувалися. DoS-атаки виявилися і в незаконних діях. Компанії можуть використовувати DoS-атаки, щоб збити своїх конкурентів на ринку. Вимагання через DoS-атаки в останні роки зросли (Pappalardo та ін. 2005). Зловмисники погрожували онлайн-бізнесу DoS-атаками та вимагали платити за захист.

Відомі атаки DoS в Інтернеті зазвичай перемагають ціль, виснажуючи її ресурси, які можуть бути будь-чими, пов'язаними з мережевими обчисленнями та продуктивністю служби, як-от пропускна здатність каналу, буфери підключення TCP, буфер додатків/служб, цикли ЦП тощо. Окремі зловмисники також можуть використовувати вразливість, зламувати цільові сервери, а потім виводити з ладу служби. Оскільки зловмисникам важко перевантажити ресурс цілі з одного комп'ютера, багато нещодавніх DoS-атак було запущено через велику кількість розподілених атакуючих хостів в Інтернеті. Ці атаки називаються розподіленими атаками на відмову в обслуговуванні (DDoS).

Під час DDoS-атаки, оскільки агрегація атакуючого трафіку може бути величезною порівняно з ресурсом жертви, атака може змусити жертву значно знизити продуктивність своїх послуг або навіть припинити надання будь-яких послуг. Порівняно зі звичайними DoS-атаками, які можна впоратися за допомогою кращого захисту систем обслуговування або заборони неавторизованого віддаленого чи локального доступу, DDoS-атаки складніші, і їм важче запобігти. Оскільки багато мимовільних хостів беруть участь у DDoS-атаках, складно розрізнити атакуючі хости та вжити проти них заходів. Останніми роками DDoS-

атаки зросли за частотою, витонченістю та серйозністю через те, що комп'ютерна вразливість швидко зростає (CERT 2006, Houle та ін. 2001), що дозволяє зловмисникам зламати та встановлювати різноманітні інструменти атаки на багатьох комп'ютерах. Бездротові мережі також страждають від атак DoS, оскільки мобільні вузли (наприклад, ноутбуки, телефони тощо) спільно використовують ті самі фізичні носії для передачі та отримання сигналів і мобільні обчислювальні ресурси (такі як пропускна здатність, центральний процесор і потужність) зазвичай більш обмежені, ніж доступні дротовим вузлам. У бездротовій мережі один зловмисник може легко підробити, змінити або ввести пакети, щоб порушити з'єднання між мобільними вузлами та спричинити ефекти DoS.

Man-in-the-Middle (MitM) атаки

Атака типу «людина посередині»(рисунок 1.5) дозволяє зловмиснику перехоплювати, надсилати та отримувати дані, призначені для когось іншого або взагалі не призначені для надсилання, без відома будь-якої сторони, поки не стане надто пізно. Атаки "людина посередині" можна скоротити багатьма способами, зокрема MITM, MitM, MiM або MIM

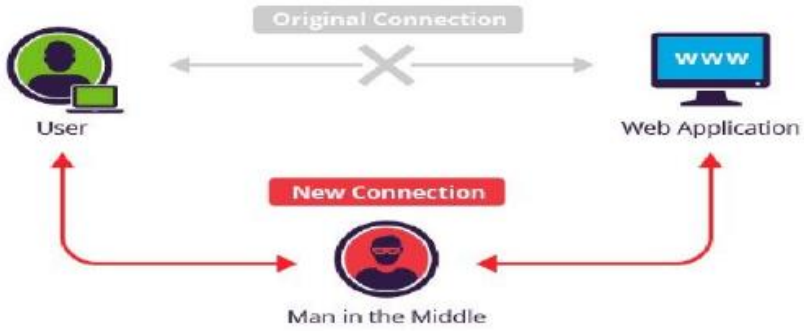


Рисунок 0.5 - Атака типу «людина посередині»

Одним із випадків атак типу "людина посередині" є динамічне прослуховування, під час якого зловмисник створює незалежні асоціації з жертвами та передає повідомлення між ними, щоб переконати їх у тому, що вони розмовляють прямо один з одним через приватну асоціацію, коли, безсумнівно, вся дискусія контролюється зловмисником. Зловмисник повинен мати здатність

перехоплювати кожне важливе повідомлення, що передається між двома жертвами, і вводити нові. Це прямо в багатьох умовах; наприклад, зловмисник у межах зони незашифрованої бездротової точки доступу (Wi-Fi) може вставити себе як людину посередині. Як атака, спрямована на обхід загальної автентифікації або дефіциту в цьому відділі, атака "людина посередині" може бути успішною, коли зловмисник може імітувати кожну прийнятну для нього кінцеву точку, що не дивно через справжні закриття. Загалом кажучи, атака MITM — це те, що можна назвати листоношею, який відкриває оголошення про ваш банк, записує ваші пам’ятки, а потім знову запечатує конверт і доставляє його до входу. Більшість криптографічних умов включають певний тип автентифікації кінцевої точки, зокрема для стійких атак MITM. Наприклад, TLS може автентифікувати одну або дві сторони за допомогою загальнодовіреного експерта з підтвердження

Ефективне виконання MITM має два чітких етапи: перехоплення та дешифрування; який передбачає перебування в межах фізичної близькості від наміченої цілі, а інший, що виключний, включає зловмисне програмне забезпечення, відоме як атака "людина в браузері" (MITB). При звичайній атаці MITM зловмиснику потрібен доступ до незахищеного або неефективно закріпленого комутатора Wi-Fi. Подібного роду асоціації в основному зустрічаються на відкритих територіях з безкоштовними точками Wi-Fi і навіть у деяких людей вдома. Зловмисник перевірить комутатор за допомогою коду, шукаючи певні недоліки, наприклад, використання секретного ключа за замовчуванням або погане використання, або прогалини в безпеці через погане розташування комутатора. Коли зловмисник виявить безсилля, він вставить свої інструменти в середину ПК клієнта та на сайти, які клієнт відвідує. Більш свіжий варіант цієї атаки набув популярності серед кіберзлочинців завдяки своїй простоті виконання. Під час атаки типу «людина в браузері» кожному зловмиснику потрібен підхід до ін’єкції зловмисного програмного забезпечення на комп’ютері, яке потім встановлюється у браузер без навчання клієнтів і записує інформацію, що надсилається. між жертвою та конкретними сайтами, зосередженими на сайтах, наприклад, фінансових установах, на яких закодовано шкідливе програмне

забезпечення. Після того, як зловмисне програмне забезпечення зібрало певну інформацію, для збору якої було модифіковано, воно потім передає цю інформацію назад зловмиснику.

Алгоритм виконання з двох кроків:

Початковий крок - перехоплення активність клієнта через систему зловмисника до того, як він досягне свого призначення. Найвідомішим (і найпростішим) методом для цього є неактивна атака, під час якої зловмисник створює вільні/відкриті точки доступу Wi-Fi; доступні широкому суспільству. Зазвичай їх називають відповідно до їхньої місцевості, вони не захищені ключовим словом. Як тільки потерпілий підключається до такої точки доступу, зловмисник отримує повну доступність для будь-якої онлайн-торгівлі інформацією. Зловмисники, які бажають застосувати більш динамічну стратегію перехоплення, можуть здійснити одну з наступних атак:

- Підробка IP-адреси полягає в тому, що зловмисник маскується під програму, змінюючи заголовки посилок в IP-адресі. Відповідно, клієнти, які намагаються отримати URL-адресу, пов’язану з програмою, спрямовуються на сайт зловмисника
- ARP-спуфінг — це спосіб зв’язати mac-адресу зловмисника з ip-адресою законного користувача в локальній мережі за допомогою підроблених повідомлень ARP. Згодом інформація, надіслана клієнтом до хосту ip-доставки, замість цього передається зловмисникові
- Підробка DNS, яку ще називають отруєнням сховища DNS, передбачає проникнення на DNS-сервер і зміну запису адреси сайту. Відповідно, клієнти, які намагаються потрапити на сайт, направляють налаштований dns-запис на сайт зловмисника.

Після перехоплення будь-яке двостороннє переміщення SSL має бути розшифровано без сповіщення клієнта чи програми. Для цього існують різні стратегії:

- Спуфінг HTTPS надсилає підтвердження самозванця у браузер жертви, коли буде зроблено початковий запит на зв’язок із безпечним сайтом. Він містить розширений відбиток пальця, пов’язаний із торгованою програмою, який браузер підтверджує відповідно до наявного списку конфіденційних місць призначення. Тоді зловмисник готовий отримати доступ до будь-якої інформації, введеної жертвою, перш ніж вона буде передана до програми.
- SSL BEAST (зловживання браузером проти SSL/TLS) зосереджується на безпорадності варіанта TLS 1.0 у SSL. Тут ПК пораненого заражено згубним JavaScript, який перехоплює зашифровані ласощі, надіслані веб-додатком. Тоді квадратний ланцюжок програми (CBC) потрапляє під загрозу для того, щоб декодувати її ліки та маркери автентифікації
- Викрадення SSL відбувається, коли зловмисник передає створені ключі автентифікації як клієнту, так і додатку під час рукостискання TCP. Це встановлює те, що здається, за всіма ознаками, безпечним зв’язком, коли насправді людина посередині контролює весь сеанс
- Вилучення SSL мінімізує асоціацію HTTPS із HTTP, перехоплюючи автентифікацію TLS, надіслану програмою клієнту. Зловмисник надсилає розшифровану форму сайту програми клієнту, зберігаючи закріплений сеанс із програмою. При цьому вся сесія клієнта помітна зловмиснику

Cross-Site Request Forgery (CSRF)

CSRF-атаки відбуваються, коли зловмисний веб-сайт змушує веб-браузер користувача виконувати небажану дію на надійному сайті. Ці атаки називають «сплячим гігантом» веб-вразливостей, оскільки багато сайтів в Інтернеті не можуть захистити від них, тому що вони здебільшого ігноруються спільнотами веб-розробників і безпеки. Атаки CSRF не згадуються в Класифікації загроз веб-безпеці та рідко обговорюються в науковій чи технічній літературі. Атаки CSRF легко діагностувати, легко використати та легко виправити. Вони існують через те, що веб-розробники не знають причин і серйозності атак CSRF. Веб-

розробники також можуть мати помилкове враження, що захист від більш відомого міжсайтового сценарію (XSS) також захистити від атак CSRF.

Атаки CSRF часто використовують механізми автентифікації цільових сайтів. Корінь проблеми полягає в тому, що веб-автентифікація зазвичай гарантує сайту, що запит надійшов із браузера певного користувача, але це не гарантує, що користувач дійсно запитав або авторизував запит. Наприклад, припустимо, що Аліса відвідує цільовий сайт «Б». «Б» передає браузеру Аліси файл cookie, що містить псевдовипадковий ідентифікатор сеансу sid, щоб відстежувати її сеанс. Алісу просять увійти на сайт, і після введення її дійсного імені користувача та пароля сайт фіксує той факт, що Аліса ввійшла до сеансу sid. Коли Аліса надсилає запит Т, її браузер автоматично надсилає файл cookie сеансу, що містить sid. Потім Т використовує свій запис, щоб визначити, що сеанс надходить від Аліси. Тепер припустімо, що Аліса відвідує шкідливий сайт «В». Вміст, наданий «В», містить код Javascript або тег зображення, який змушує браузер Аліси надсилати HTTP-запит до «Б». Оскільки запит спрямовується до «Б», браузер Аліси «корисно» додає файл cookie сеансу sid до запиту. Побачивши запит, «Б» робить висновок за наявністю файлу cookie, що запит надійшов від Аліси, тому «Б» виконує запитану операцію з обліковим записом Аліси. Це успішна атака CSRF.

Більшість інших механізмів веб-автентифікації страждають від тієї ж проблеми. Наприклад, механізм HTTP BasicAuth змусить Алісу повідомити своєму браузеру своє ім’я користувача та пароль для сайту «Б», а потім браузер «допоміжно» приєднає ім’я користувача та пароль до майбутніх запитів, надісланих «Б». Крім того, «Б» може використовувати сертифікати SSL зі сторони клієнта, але така ж проблема виникне, оскільки браузер «корисно» використовуватиме сертифікат для виконання запитів до сайту «Б». Подібним чином, якщо «Б» автентифікує Алісу за її IP-адресою, можливі CSRF-атаки. Загалом, щоразу, коли автентифікація відбувається неявно — через сайт, на який надсилається запит і з якого браузера він надходить, — існує небезпека атак CSRF. У принципі, цю небезпеку можна було б усунути, вимагаючи від користувача виконувати явні дії, які неможливо підробити (наприклад, повторне введення

імені користувача та пароля) для кожного запиту, надісланого на сайт, але на практиці це спричинило б серйозні проблеми з користуванням. Найбільш стандартні та широко використовувані механізми автентифікації не можуть запобігти атакам CSRF, тому практичне рішення слід шукати в іншому місці.

Окремі сайти, а також фреймворки можуть захистити себе від атак CSRF, дотримуючись таких запобіжних заходів:

1. Дозвольте запитам GET лише отримувати дані, а не змінювати дані на сервері. Ця зміна захищає сайти від атак CSRF за допомогою тегів <img> або інших типів запитів GET. Крім того, ця рекомендація відповідає RFC 2616 (HTTP/1.1): зокрема, було встановлено конвенцію про те, що методи GET і HEAD не повинні мати значення виконання дій, окрім отримання. Ці методи слід вважати «безпечними». Хоча цей захист сам по собі не запобігає CSRF (оскільки зловмисники можуть використовувати запити POST), його можна поєднати, щоб повністю запобігти вразливостям CSRF.

2. Вимагайте, щоб усі запити POST включали псевдовипадкове значення. Коли користувач відвідує сайт, сайт має генерувати (криптографічно надійне) псевдовипадкове значення та встановлювати його як файл cookie на комп’ютері користувача. Сайт повинен вимагати, щоб кожне надсилання форми включало це псевдовипадкове значення як значення форми, а також як значення cookie. Коли на сайт надсилається запит POST, запит слід вважати дійсним, лише якщо значення форми та значення cookie збігаються. Коли зловмисник надсилає форму від імені користувача, він може лише змінювати значення форми. Зловмисник не може читати будь-які дані, надіслані із сервера, або змінювати значення файлів cookie відповідно до політики того самого джерела (див. Додаток В). Це означає, що хоча зловмисник може надіслати будь-яке значення з формою, він не зможе змінити або прочитати значення, що зберігається в файлі cookie. Оскільки значення cookie та значення форми мають бути однаковими, зловмисник не зможе успішно надіслати форму, якщо він не зможе вгадати псевдовипадкове значення.

3. Використовуйте незалежне псевдовипадкове значення облікового запису користувача. Псевдовипадкові значення, пов’язані з значеннями користувача не зможуть запобігти атаці «Login CSRF».

1.2. Основи криптографічного захисту даних

Симетричне шифрування

У шифруванні з симетричним ключем будь-який користувач, який використовує систему шифрування, має копію єдиного секретного ключа, цей секретний ключ використовується для шифрування та дешифрування, це швидше, ніж асиметричне шифрування, у симетричному шифруванні відправник ділиться тим самим секретним ключем з одержувачем, щоб використати його в процесі дешифрування. Оскільки секретний ключ спільний для відправника та одержувача, використовувати його стає ризиковано. У шифруванні з симетричним ключем рекомендовано використовувати ключ довжиною 128 біт, щоб уникнути або ускладнити будь-яку спробу злому. Деякі з найпоширеніших алгоритмів шифрування: AES, RC4, DES, 3DES, RC5 і RC6. З цих алгоритмів найкращим є AES. Симетричне шифрування використовується, коли потрібна швидкість процесу над безпекою. Деякі з найпоширеніших способів використання симетричного шифрування включають:

- Банківські та фінансові організації (шифрування інформації кредитної картки та ідентифікаційної інформації (PII) для завершення транзакцій).
- Зберігання даних (шифрування даних у спокої)

DES один із найстаріших методів симетричного шифрування, він був запроваджений у 1976 році. Він використовує 56-бітний ключ шифрування, DES перетворює 64-бітний відкритий текст блоку на зашифрований, розділяючи блок на дві окремі частини по 32 біти. DES більше не використовується з міркувань безпеки. У 2005 р. недоліком DES була мала довжина ключа, який можна було

легко зламати за допомогою атаки грубої сили, щоб також зламати його, нова версія TLS (безпека транспортного рівня) більше не залежить від DES.

3DES (потрійний Симетричний алгоритм шифрування даних) це оновлена версія алгоритму DES. Він застосовує DES тричі для кожного блоку даних, що ускладнює його зламати. Він широко використовується в платіжних системах і фінансовій галузі, також стає частиною криптографічних протоколів, таких як TLS, SSH, IPsec і OpenVPN. Було виявлено багато дірок у безпеці. Національний інститут стандартів і технологій (NIST) не схвалив використання цього в проекті керівництва в 2019 році, TLS 1.3. Остання версія протоколів SSL/TLS припинила використання 3DES.

AES (Advanced Encryption System). Він також відомий як Rijndael. AES став стандартом шифрування після схвалення NIST у 2001 році. Він перетворює відкритий текст на блоки, а потім застосовується шифрування. AES набагато швидший, ніж DES. AES має багато переваг, наприклад, це безпечний, швидкий, гнучкий і багаторазовий варіант ключа. AES широко використовується в багатьох програмах, таких як бездротова безпека, безпека процесора, протоколи SSL/TLS, шифрування мобільних програм і VPN . Багато державних установ, таких як Агентство національної безпеки (NSA), залежать від алгоритму шифрування AES для захисту та захисту конфіденційних даних.

Асиметричне шифрування

Воно також відоме як криптографія з відкритим ключем. Вона включає кілька ключів для шифрування та дешифрування, і використовує два відмінні ключі шифрування, пов’язані один з одним, перший ключ відомий як відкритий ключ, а інший ключ відомий як приватний ключ. Відкритий ключ доступний кожному, хто хоче надіслати повідомлення відправнику. Другий приватний ключ залишається секретним, його знає лише відправник. Повідомлення, зашифроване відкритим ключем, можна розшифрувати за допомогою закритого ключа. Повідомлення, зашифроване приватним ключем, може бути розшифровано за допомогою відкритого ключа. Безпека відкритого ключа не потрібна, оскільки він доступний і може передаватись через Інтернет. Асиметричне шифрування

вважається найкращим вибором для передачі інформації. Асиметричне шифрування використовується в моделі зв’язку між клієнтом і сервером, сертифікат створюється для визначення місцезнаходження сервера, він містить профіль організації та інформацію. SSL/TLS використовує асиметричне та симетричне шифрування. Усі комунікації використовують лише відкриті ключі, а закритий ключ не передається. Деякі з алгоритмів використовують техніку RSA, яка використовується для шифрування й автентифікації, і Pretty Good Privacy (PGP), яка використовується для захисту електронних листів. Асиметричне шифрування використовується, коли безпека має пріоритет над швидкістю. Найпоширенішим використанням асиметричного шифрування є:

- Цифровий підпис(використовується для підтвердження підпису особи)
- Blockchain(підтвердження особи для авторизації транзакції криптовалюти)
- Інфраструктура відкритих ключів(авторизація ключів шифрування шляхом видачі цифрових сертифікатів)

Алгоритм асиметричного шифрування RSA

Представлений у 1977 році, він вважається найпоширенішим асиметричним алгоритмом шифрування, він використовує два великих випадкових простих числа, ці числа перемножуються, щоб отримати ще одне величезне число. Головоломка полягає в тому, щоб знайти оригінальні прості числа. Якщо використовується правильна довжина, для злому ключа RSA-768, який є нижчим за стандартний 2048-бітний RSA, потрібно не менше 1500 років. Використання RSA має кілька довжин, наприклад 768-біт, 1024-біт, 2048-біт і 4096-біт, що забезпечує безпеку та масштабованість RSA. RSA може працювати з різними програмами та протоколами, включаючи PKI та SSL/TLS.

Алгоритм асиметричного шифрування ECC

Це ще один тип асиметричного алгоритму шифрування. Він використовує складну математичну концепцію, засновану на еліптичних кривих над кінцевим полем. ECC має приватний ключ і відкритий ключ, відкритий ключ

використовується для перевірки процесу, підписаного приватним ключем, а приватний ключ використовується для розшифровки даних, які шифруються за допомогою відкритого ключа. ECC забезпечує найкращий захист від сучасних методів злому. Він забезпечує той самий рівень захисту RSA, але з меншою довжиною ключа. ECC - це швидша продуктивність, а також менше навантаження на мережу та комп'ютерні системи. Якщо використовувати ECC із сертифікатом SSL/TLS, це скорочує час, необхідний для рукостискань SSL/TLS, а також шифрування ECC використовується для шифрування програм. Недоліком ECC є те, що багато серверного програмного забезпечення ще не підтримують ECC для сертифіката SSL/TLS.

Хешування та цифрові підписи

Цифровий підпис - різновид електронного підпису. Це математичний алгоритм, який зазвичай використовується для перевірки автентичності та цілісності повідомлення (наприклад, електронного листа, транзакції кредитної картки або цифрового документа). Цифрові підписи створюють віртуальний відбиток пальця, який є унікальним для особи чи організації та використовується для ідентифікації користувачів і захисту інформації в цифрових повідомленнях або документах. У електронних листах сам вміст електронної пошти стає частиною цифрового підпису. Цифрові підписи значно безпечніші, ніж інші форми електронних підписів.

Хеш-функція - це рядок цифр і літер фіксованої довжини, згенерований за допомогою математичного алгоритму та файлу довільного розміру, наприклад електронного листа, документа, зображення або іншого типу даних. Цей згенерований рядок є унікальним для файлу, який хешується, і є односторонньою функцією - обчислений хеш не можна змінити, щоб знайти інші файли, які можуть генерувати те саме хеш-значення. Деякі з найпопулярніших алгоритмів хешування, які використовуються сьогодні, це Secure Hash Algorithm-1 (SHA-1), Secure Hash Algorithm-2 (SHA-2 і SHA-256) і Message Digest 5 (MD5).

Інфраструктура відкритих ключів (PKI) - складається з політик, стандартів, людей і систем, які підтримують розповсюдження відкритих ключів і перевірку ідентичності осіб або організацій за допомогою цифрових сертифікатів і центру сертифікації.

Центр сертифікації (ЦС) - це довірена третя сторона, яка перевіряє особу особи та генерує пару відкритий/приватний ключ від її імені або пов'язує існуючий відкритий ключ, наданий цією особою. Після того, як центр сертифікації підтверджує чиюсь особу, він видає цифровий сертифікат, який має цифровий підпис ЦС. Цифровий сертифікат потім можна використовувати для перевірки особи, пов'язаної з відкритим ключем, коли це буде потрібно.

Цифрові сертифікати - аналогічні водійським правам, оскільки вони призначені для ідентифікації власника сертифіката. Цифрові сертифікати містять відкритий ключ особи чи організації та мають цифровий підпис ЦС. У сертифікат також можна включити іншу інформацію про організацію, особу та ЦС.

Pretty Good Privacy (PGP)/OpenPGP - є альтернативою PKI. За допомогою PGP/OpenPGP користувачі «довіряють» іншим користувачам, підписуючи сертифікати людей, які можна перевірити. Чим більше взаємопов'язані ці підписи, тим вища ймовірність верифікації конкретного користувача в Інтернеті. Ця концепція називається «Мережа довіри».

Цифрові підписи доводять, що цифрове повідомлення або документ не було змінено, навмисно чи ненавмисно, з моменту його підписання. Цифрові підписи роблять це, генеруючи унікальний хеш повідомлення або документа та шифруючи його за допомогою закритого ключа відправника. Згенерований хеш є унікальним для повідомлення чи документа, і зміна будь-якої його частини повністю змінить хеш.

Після завершення повідомлення або цифрового документу, він підписується цифровим підписом і надсилається одержувачу. Потім одержувач створює власний хеш повідомлення або цифрового документа та розшифровує хеш відправника (включений у вхідне повідомлення) за допомогою відкритого ключа відправника. Одержувач порівнює хеш, який вони генерують, із

розшифрованим хешом відправника; якщо вони збігаються, це означає, що повідомлення або цифровий документ не було змінено, а відправника автентифіковано.

Використання цифрових підписів у поєднанні з PKI або PGP посилює їх і зменшує можливі проблеми безпеки, пов’язані з передачею відкритих ключів, шляхом перевірки того, що ключ належить відправнику, і перевіряючи особу відправника. Безпека цифрового підпису майже повністю залежить від того, наскільки добре захищено закритий ключ. Без PGP або PKI неможливо підтвердити чиюсь особу або відкликати зламаний ключ; це може дозволити зловмисникам видавати себе за когось без жодного методу підтвердження.

Завдяки використанню довіреної третьої сторони цифрові підписи можна використовувати для ідентифікації та перевірки осіб і забезпечення цілісності повідомлення.

Оскільки безпаперова онлайн-взаємодія використовується ширше, цифрові підписи можуть допомогти вам захистити цілісність ваших даних. Розуміючи та використовуючи цифрові підписи, ви можете краще захистити свою інформацію, документи та транзакції.

Сертифікати безпеки SSL/TLS

Незалежно від того, звідки ви отримуєте доступ до своїх хмарних, мобільних або веб-додатків, з’єднання між вашим пристроєм і будь-якою іншою точкою можна маршрутизувати через десятки незалежних мережевих систем. За допомогою стеження, спуфінгу та інших форм інтернет-прослуховування неавторизовані особи можуть ідентифікувати та викрасти особисті дані та іншу конфіденційну інформацію, таку як номери кредитних карток, інформацію про клієнтів і партнерів і PIN-коди.

Протокол Secure Sockets Layer (SSL) був розроблений для конфіденційної та безпечної передачі інформації через Інтернет. SSL розташовується під такими протоколами додатків, як HTTP, SMTP, POP3 і FTP(рисунок 1.6), і вище протоколу підключення TCP/IP. Використовується методом доступу HTTPS. Без SSL єдиною безпекою є автентифікація клієнта сервером за допомогою імені користувача та

пароля. Зв’язок між клієнтом і сервером є відкритим і незахищеним. На рисунку показано різницю між незахищеним HTTP-запитом і безпечним SSL-запитом.

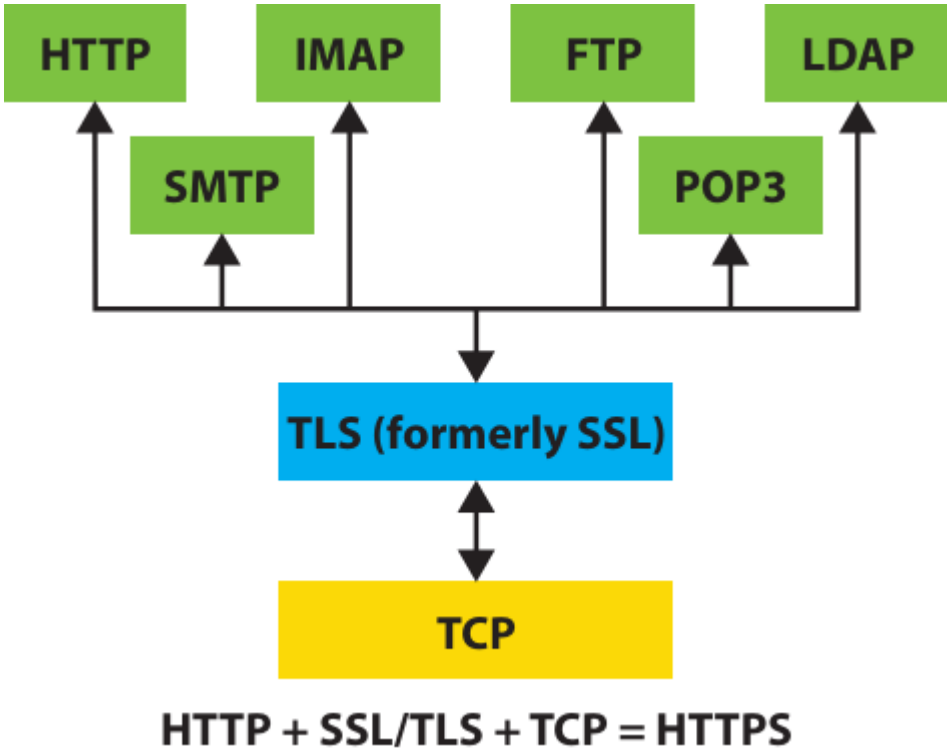


Рисунок 0.6 - веб протоколи

У той час як Transport Layer Security (TLS) є наступником протоколу Secure Sockets Layer (SSL) і домінуючим галузевим стандартом; обидва вони є криптографічними протоколами, які забезпечують безпечний зв’язок в Інтернеті для таких речей, як перегляд веб-сторінок, доступ до хмари, зв’язок електронною поштою, обмін миттєвими повідомленнями та інші передачі даних. Існують невеликі відмінності у функціональності між SSL і TLS, і, завідомо, SSL є менш безпечним, ніж поточний TLS, але протоколи залишаються практично незмінними. Через ці численні вразливості в SSL Symantec надає ретроактивну підтримку SSL, але наполегливо рекомендує всім клієнтам перевести свої веб-сайти та сервери на підтримку з’єднань через TLS1.1 або 1.2. З метою спрощення та з огляду на знайомство ми будемо використовувати термін «SSL» у цьому документі, коли йдеться про ці колективні стандарти. Перевагою SSL є безпечне автентифіковане спілкування через загальнодоступні чи приватні мережі. Зростаючий недолік SSL полягає в тому, що він створює «сліпі зони» в мережевій інфраструктурі через його основну мету - захист зв’язку. Оскільки вміст і дані в

трафіку SSL залишаються прихованими для більшості інструментів мережевої безпеки, ризик витоку даних, викрадання та крадіжки вищий. Традиційні та навіть сучасні пристрої мережевої безпеки, такі як брандмауери наступного покоління (NGFW), системи запобігання вторгненням (IPS), запобігання втраті даних (DLP) і рішення для захисту від зловмисного програмного забезпечення/пісочниці, неефективні для виявлення та блокування розширених програм, які використовують SSL для приховування.

1.3. Моделі безпеки веб-застосунків

Модель загроз (Threat Model)

Через обмежені ресурси власника системи важко пом’якшити кожну вразливість у системі. Тому власники систем повинні визначати пріоритетність ризиків і відповідним чином ставитися до них. Ключовим кроком у визначенні ризику є виявлення загрозливих подій, які впливають на ймовірність і вплив ризику. Подія загрози стосується будь-якої події, під час якої суб’єкт загрози1 за допомогою вектора загрози2 діє проти активу таким чином, що може завдати шкоди. У контексті кібербезпеки загрозливі події можна охарактеризувати тактикою, технікою та процедурами (TTP), які використовують суб’єкти загрози. Моделювання загроз допомагає власникам всебічно ідентифікувати події загроз, які стосуються системи, щоб власники могли зосередитися на впровадженні ефективних заходів контролю для захисту ключових компонентів у системі. Це ускладнює для супротивника скомпрометувати ключові компоненти, закріплюючись, повертаючись і пересуваючись убік у системі. За допомогою моделі загрози власники системи також можуть уникнути сліпих зон у виявленні подій загрози.

Користувачі можуть підходити до аналізу загроз на трьох різних рівнях: з точки зору управління, з точки зору системи та з точки зору обладнання або програми. Нижче наведено загальний опис кожного рівня.

- Рівень управління (поза сферою дії) - канали розвідки та тенденції аналізуються з акцентом на зовнішні фактори, таких як геополітика. Аналітики зазвичай проводять профілювання супротивника, враховуючи широкі мотиви та дії супротивника до та після вторгнення. Організації роблять це з точки зору високого рівня, як правило, для споживання керівництвом. Прикладом такого підходу є загальна структура кіберзагроз, яку використовує Управління директора національної розвідки США (ODNI), яка розглядає чотири горизонтальні етапи «життєвого циклу загрози»: підготовку, залучення, присутність і наслідки, розширюючи при цьому вертикалі цілей, дій та індикаторів.
- Системний рівень (за обсягом) - цей підхід розглядає системні конструкції та зв’язки, а також поведінку системи у формі компонентів, архітектурних рівнів і потоків даних. Користувач спочатку моделює активи, потоки даних і межі в середовищі, перш ніж визначати відповідні загрози для системи. У цьому документі ми зосередимося на системному рівні.
- Рівень обладнання або програми (поза межами) – аналіз загроз на цьому рівні є найбільш детальним. Це часто передбачає полювання на кіберзагрози, кореляцію журналів, детальне сортування даних, розширену аналітику та евристичні методи тощо для детального сканування на предмет уникнення та використання опублікованих уразливостей.

Як згадувалося, не існує єдиного правильного чи неправильного підходу до моделювання загроз. Користувачі повинні вибрати підхід на основі контексту бізнес-потреб, корпоративного середовища, ситуації та аудиторії.

Оцінка ризиків

Оцінка ризиків є одним із фундаментальних компонентів процесу управління організаційними ризиками. Оцінка ризику використовується для виявлення, оцінки та визначення пріоритетності ризику для організаційних операцій (тобто місії, функцій, іміджу та репутації), організаційних активів, окремих осіб, інших організацій та країни, що є результатом роботи та використання інформаційних систем. Метою оцінки ризиків є інформування осіб,

які приймають рішення, та підтримка реагування на ризики шляхом виявлення: відповідних загроз для організацій або загроз, спрямованих через організації проти інших організацій; уразливості як внутрішні, так і зовнішні для організацій; вплив (тобто шкода) на організації, який може виникнути з огляду на потенціал загроз, що використовують уразливості; і ймовірність заподіяння шкоди. Кінцевим результатом є визначення ризику (тобто, як правило, функція ступеня шкоди та ймовірності заподіяння шкоди). Оцінку ризиків можна проводити на всіх трьох рівнях ієрархії управління ризиками, включаючи рівень 1 (рівень організації), рівень 2 (рівень місії/бізнес-процесу) і рівень 3 (рівень інформаційної системи). На рівнях 1 і 2 організації використовують оцінки ризиків, щоб оцінити, наприклад, системні ризики, пов’язані з інформаційною безпекою, пов’язані з діяльністю управління та управління організацією, місією/бізнес-процесами, архітектурою підприємства або фінансуванням програм інформаційної безпеки. На рівні 3 організації використовують оцінку ризиків, щоб ефективніше підтримувати впровадження структури управління ризиками (тобто, категоризація безпеки; вибір, впровадження та оцінка засобів контролю безпеки; інформаційна система та авторизація загального контролю; і моніторинг контролю безпеки) і саме цей рівень відповідає за оцінку ризиків у веб застосунку.

1.4. Регулятивні вимоги та стандарти безпеки

Огляд GDPR та його вимоги до захисту даних

Загальний регламент захисту даних (GDPR) застосовується з 25 травня 2018 року. Він має загальне застосування до обробки персональних даних у ЄС, встановлюючи більш широкі зобов'язання для контролерів даних і обробників, а також забезпечуючи посилений захист для суб'єктів даних(рисунок 1.7). Незважаючи на те, що GDPR безпосередньо застосовується як закон у всіх державах-членах, він дозволяє надати певним питанням подальшої чинності в національному законодавстві. В Ірландії національним законом, який, серед іншого, надає подальшу дію GDPR, є Закон про захист даних 2018 року. Що є персональними даними? GDPR визначає «персональні дані» як будь-яку інформацію, що стосується особи, яку можна ідентифікувати, яку можна прямо чи опосередковано ідентифікувати, зокрема за допомогою посилання на ідентифікатор. Це визначення передбачає широкий спектр особистих ідентифікаторів, які становлять персональні дані, включаючи ім'я, ідентифікаційний номер, дані про місцезнаходження або онлайн-ідентифікатор, що відображає зміни в технології та спосіб, у який організації збирають інформацію про людей.

Контролери даних — це особа чи організація, які (одноосібно чи разом з іншими) визначають цілі та спосіб, у який будь-які персональні дані обробляються чи мають оброблятися. Контролер даних може бути єдиним контролером даних або спільним контролером даних з іншою особою чи організацією. Однак, якщо послуги надаються безпосередньо приватною лікарнею, добровільними лікарнями, агентствами чи приватними підрядниками, приватна лікарня, добровільна лікарня, агентство чи приватний підрядник можуть бути контролером даних.

Обробники даних — це ті, хто обробляє персональні дані від імені контролера. Це не включає співробітника контролера, який обробляє дані під час

роботи. Обробник даних може бути притягнутий до відповідальності, якщо він відповідальний за порушення захисту даних.

Рисунок 0.7 – структура GDRP

Основними принципами GDPR є:

- Персональні дані повинні оброблятися прозоро
- Компанія повинні мати конкретну мету для збору ваших даних
- Компанія можемо зберігати дані лише стільки часу, скільки необхідно для досягнення мети, для якої вони були використані зібрані. Ми видаляємо медичні записи відповідно до наших Правил зберігання записів політика.
- Якщо дані зберігаються на комп’ютерах, ми повинні гарантувати, що ці комп’ютери та мережі безпечні
- Якщо дані знаходяться в паперовому форматі, ми зобов’язані забезпечити їх безпеку як комп'ютерний запис

Особисті дані можуть бути поширені по всій організації - у базах даних, сховищах даних, хмарних середовищах, багатокористувацьких серверах, внутрішніх жорстких драйверах, застарілих операційних системах і навіть за межами організації. Це означає, що відповідність GDPR передбачає два типи інвентаризації: визначення всіх різноманітних технологій, систем і програм (цифрових, фізичних або комбінованих), де записуються та зберігаються персональні дані, а також визначення всіх відповідних даних і даних, що входять до сфери дії, до яких застосовуватимуться принципи GDPR. Останнє, очевидно, стосується персональних даних, що належать громадянам ЄС, які ваша організація повинна оцінити за кількома ключовими питаннями відповідності(рисунок 1.8):

- Чому дані були зібрані?
- Як збиралися дані?
- Хто використовує дані та чи передаються вони третім особам?
- Як довго ви будете зберігати дані?
- Наскільки добре дані захищені від несанкціонованого доступу?

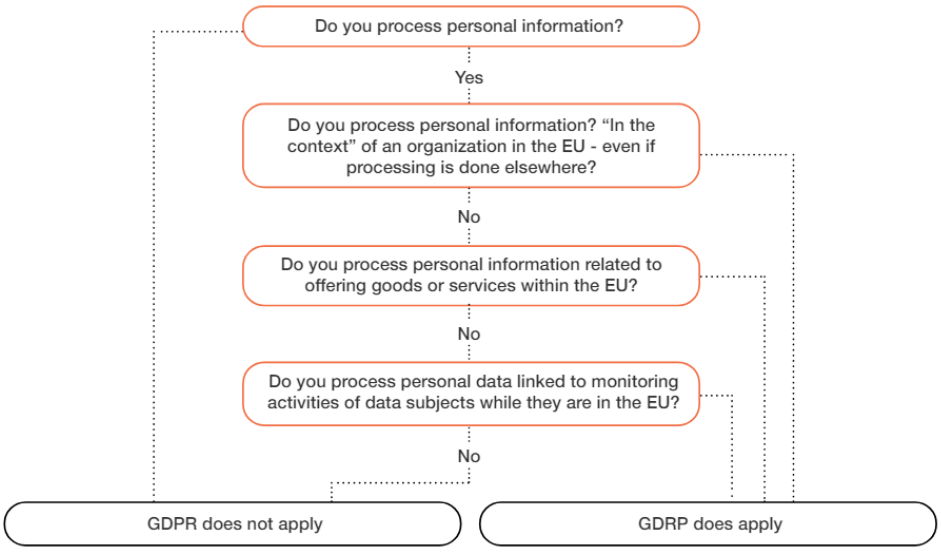


Рисунок 0.8 - Коли потрібен GDPR?

Важливо те, що ці два типи інвентаризації нерозривно пов’язані між собою для досягнення відповідної ефективності. Що робить ефективне дотримання

актуальним, так це те, що GDPR передбачає фінансові штрафи в розмірі до 4 відсотків фінансового доходу групи або 20 мільйонів євро за порушення.

GDPR поширюється на всіх, хто веде бізнес в ЄС або з ЄС, навіть якщо вони там не перебувають, включаючи Великобританію. Організації ЄС повинні будуть враховувати ризики передачі даних країнам, що не входять до ЄС, і міжнародним процесорам, які можуть не відповідати вимогам. Будь-яка особа, яка обробляє персональні дані європейських громадян, незалежно від того, чи знаходиться ця особа/організація чи обробка даних у ЄС, підпадає під дію правил GDPR.

PCI DSS для обробки платіжних даних

Кажуть, що американський злочинець двадцятого століття Віллі Саттон грабує банки, тому що «саме там гроші». Така сама мотивація в нашу цифрову епоху робить постачальників товарів та послуг новою мішенню для фінансового шахрайства. Іноді слабка безпека деяких продавців дає зловмисникам змогу легко викрадати та використовувати особисту фінансову інформацію споживача з транзакцій платіжних карток і систем обробки.

Це серйозна проблема: за даними PrivacyRights, у період з січня 2005 року по липень 2018 року було зламано понад 10,9 мільярда записів із конфіденційною інформацією. Оскільки постачальник є ключовим учасником операцій з платіжними картками, йому необхідно використовувати стандартні процедури безпеки та технології, щоб запобігти крадіжці даних власника картки.

Уразливості продавця можуть з’явитися майже будь-де в екосистемі обробки карток, зокрема:

- торгові пристрої;
- мобільні пристрої, персональні комп'ютери або сервери;
- бездротові точки доступу;
- програми для веб-покупок;
- паперові системи зберігання;
- передача даних власника картки постачальникам послуг;

- у підключеннях віддаленого доступу.

Уразливості також можуть поширюватися на системи, якими керують еквайери, тобто фінансові установи, які ініціюють і підтримують відносини з продавцями, які приймають платіжні картки (рисунок 1.9). Відповідність PCI DSS допомагає зменшити ці вразливості та захистити дані власників карток.

Рисунок 0.9 - фінансові установи і засоби, які приймають платежі

Є три кроки для дотримання PCI DSS:

- Оцінка - визначення всіх місць розташування даних власників карток, проведення інвентаризації ваших IT-активів і бізнес-процесів для обробки платіжних карток і їх аналіз на наявність вразливостей, які можуть викрити дані власників карток.
- Ремонт - усунення виявлених уразливостей, безпечне видалення будь-яких непотрібних даних власника картки та впровадження безпечних бізнес-процесів.
- Звіт - документування деталей оцінки та виправлення, а також подання звітів про відповідність банку-еквайеру та брендам карток, з якими ви працюєте (або іншій організації, яка надіслала запит, якщо ви є постачальником послуг).

PCI DSS глобально застосовується до всіх суб’єктів, які зберігають, обробляють або передають дані власників карток та/або конфіденційні дані автентифікації. Він і відповідні стандарти безпеки адмініструються Радою стандартів безпеки PCI, яка була заснована American Express, Discover Financial Services, JCB International, MasterCard Worldwide і Visa Inc. Організації-учасники

включають продавців, банки-емітенти платіжних карток, обробників платежів, розробників та інших постачальників.

Метою стандарту безпеки даних PCI (рисунок 1.10) є захист даних власників карток і конфіденційних даних автентифікації, де б вони не оброблялися, зберігалися або передавалися.

Рисунок 0.10 - PCI безпека

Контроль безпеки та процеси, обов'язкові за PCI DSS, є життєво важливими для захисту всіх даних рахунків платіжних карток, включаючи PAN - основний номер рахунку, надрукований на лицьовій стороні платіжної картки. Продавці, постачальники послуг та інші організації, залучені до обробки платіжних карток, ніколи не повинні зберігати конфіденційні дані автентифікації після авторизації. Це включає 3- або 4-значний код безпеки(рисунок 1.11), надрукований на лицьовій або зворотній стороні картки, дані, що зберігаються на магнітній смузі чи чіпі картки (також звані «Повні дані»), а також персональні ідентифікаційні номери (PIN-код), введені власником картки.

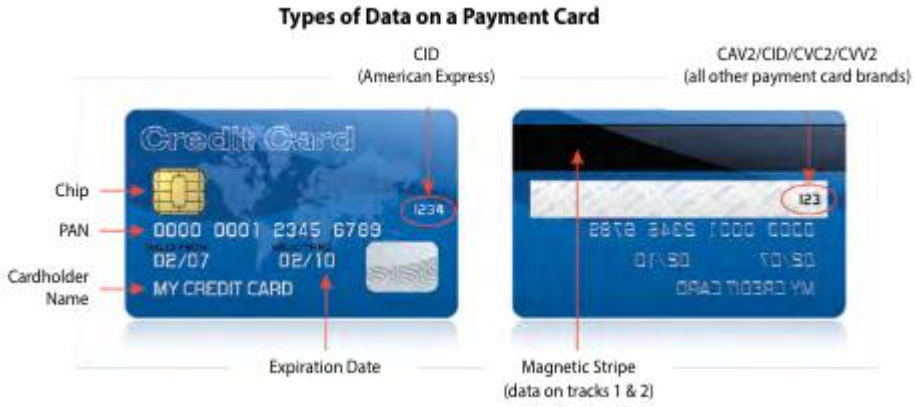


Рисунок 0.11 - елементи платіжної карти

ISO/IEC 27001

ISO/IEC 27001 - це міжнародний стандарт для впровадження системи управління інформаційною безпекою (ISMS), опублікований Міжнародною організацією зі стандартизації (ISO) і Міжнародною електротехнічною комісією (IEC). ISO випускає стандарти системи менеджменту, щоб допомогти організаціям організувати бізнес-процеси та процедури для досягнення конкретних цілей. Стандарт ISO/IEC 27001 дозволяє організаціям захищати конфіденційні дані та зменшувати ризики кібератак, викладаючи набір загальноприйнятих процедур управління та заходів безпеки інформації.

Стандарт ISO/IEC 27001 широко використовується як набір найкращих практик(рисунок 1.12), які організації можуть використовувати під час створення системи управління інформаційною безпекою. При правильному застосуванні ці найкращі методи можуть допомогти покращити стан інформаційної безпеки та встановити довіру у клієнтів. Деякі організації можуть вибрати сертифіковану СУІБ і отримати оцінку СУІБ стороннім аудитором. Незважаючи на те, що сертифікація ISO/IEC 27001 не є обов’язковою, клієнти цінують її як засіб перевірки засобів контролю інформаційної безпеки та процедур управління в організації. Сертифікаційні аудити ISO/IEC 27001 проводяться кожні три роки, однак організації також повинні проходити щорічні наглядові аудити між роками сертифікації.

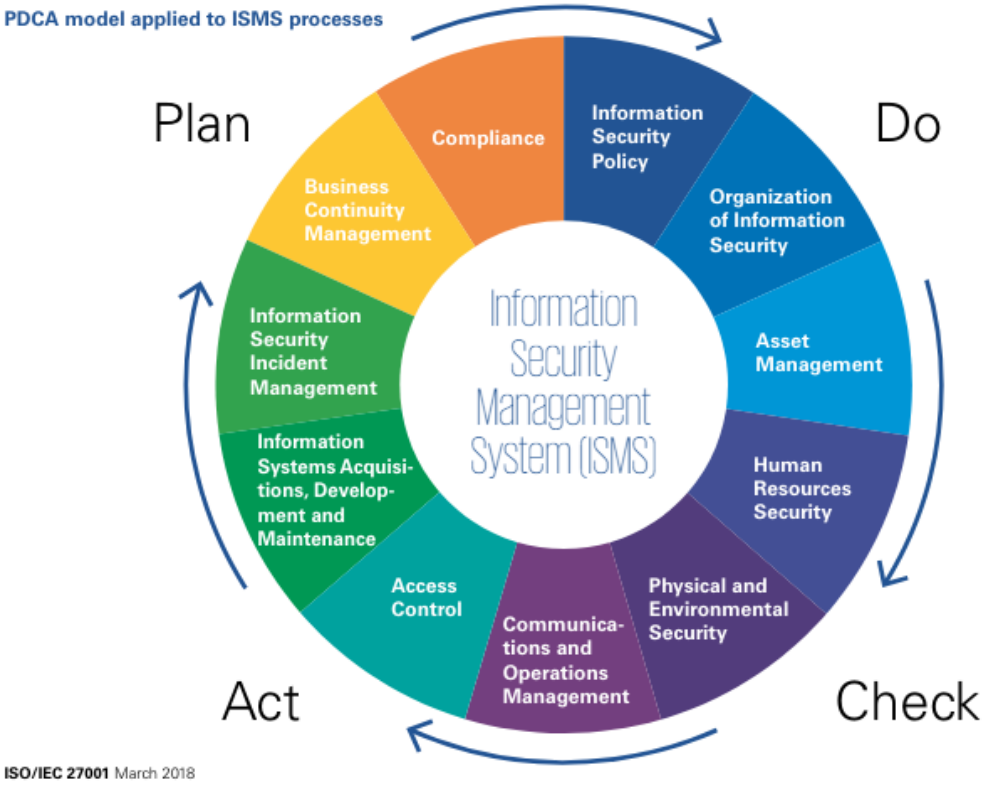


Рисунок 0.12 - ISO/IEC 27001 життєвий цикл, практики

Сертифікація ISO/IEC 27001(рисунок 1.13) служить лінзою в середовищі інформаційної безпеки організації. У поєднанні із заявою компанії про застосовність (SoA) клієнти або потенційні клієнти можуть бути впевнені, що існують основні процедури та засоби контролю для захисту їхніх даних за допомогою офіційної системи управління інформаційною безпекою.

Рисунок 0.13 – ISO/IEC сертифікація

Щоб отримати сертифікат ISO/IEC 27001, система управління інформаційною безпекою організації повинна відповідати критеріям, встановленим пунктами управління, визначеними стандартом ISO/IEC 27001. На додаток до положень про управління існує 114 засобів контролю інформаційної безпеки, які можна включити або опустити в залежності від ризиків, з якими стикається організація. Організації повинні завершити оцінку ризиків або аналіз прогалин, щоб визначити ці ризики та, у свою чергу, задокументувати обґрунтування включення/пропуску в заяві про застосовність. І сертифікація, і заява про застосовність є важливими для розуміння заходів безпеки, яких вжила організація.

Висновки

У цьому розділі ми розглянули основи безпеки веб-додатків, включаючи основні типи кібератак, основи криптографічного захисту даних і моделі безпеки веб-додатків, а також юридичні вимоги та стандарти безпеки. Ми визначили та детально проаналізували різні типи кібератак, включаючи SQL-ін’єкцію, міжсайтовий сценарій (XSS), відмову в обслуговуванні (DoS), атаки Man-in-the-Middle (MitM) і підробку міжсайтових запитів. (CSRF). Для кожного типу атаки наведено приклади, описано механізм виконання та можливі наслідки для безпеки веб-додатку. Це допомагло нам зрозуміти важливість виявлення таких загроз і захисту від них під час розробки та експлуатації веб-додатків.

Потім ми розглянули основи криптографічного захисту даних, включаючи симетричне та асиметричне шифрування, хешування та цифрові підписи, а також сертифікати безпеки SSL/TLS. Ми проаналізували, як кожна з цих технологій допомагає захистити дані від несанкціонованого доступу та зберегти цілісність даних. У розділі про моделі безпеки веб-додатків ми вивчали Threat Modeling та Risk Assessment, які дозволяють визначити потенційні загрози та вразливості в системі. Це важливий крок у побудові ефективної стратегії захисту інформації та мінімізації ризиків.

У останньому підрозділі ми проаналізували юридичні вимоги та стандарти безпеки, такі як GDPR, PCI DSS та ISO/IEC 27001. Ці стандарти та правила визначають вимоги до захисту даних і допомагають організаціям відповідати юридичним вимогам і забезпечувати високий рівень захисту інформації. Ми розробили всебічне розуміння основ безпеки веб-додатків, які необхідні для забезпечення їх надійної роботи та захисту даних користувачів.

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ У СФЕРІ КІБЕРБЕЗПЕКИ

2.1. Популярні інструменти для захисту веб-застосунків

WAF (Web Application Firewalls)

Брандмауер веб-додатків (WAF) забезпечує захист веб-додатків для онлайн-сервісів від зловмисних атак безпеки, таких як впровадження SQL, міжсайтовий сценарій (XSS). Він застосовує набір правил до розмови HTTP2(рисунок 2.1). WAF забезпечує захист веб-серверів. Спілкування веб-додатків відбувається за моделлю клієнт-сервер, де сервер надсилає повідомлення лише у відповідь на запит клієнта. Типовий брандмауер fi захищає мережевий рівень, тоді як брандмауер веб-додатків захищає сервер і весь його стек програм.

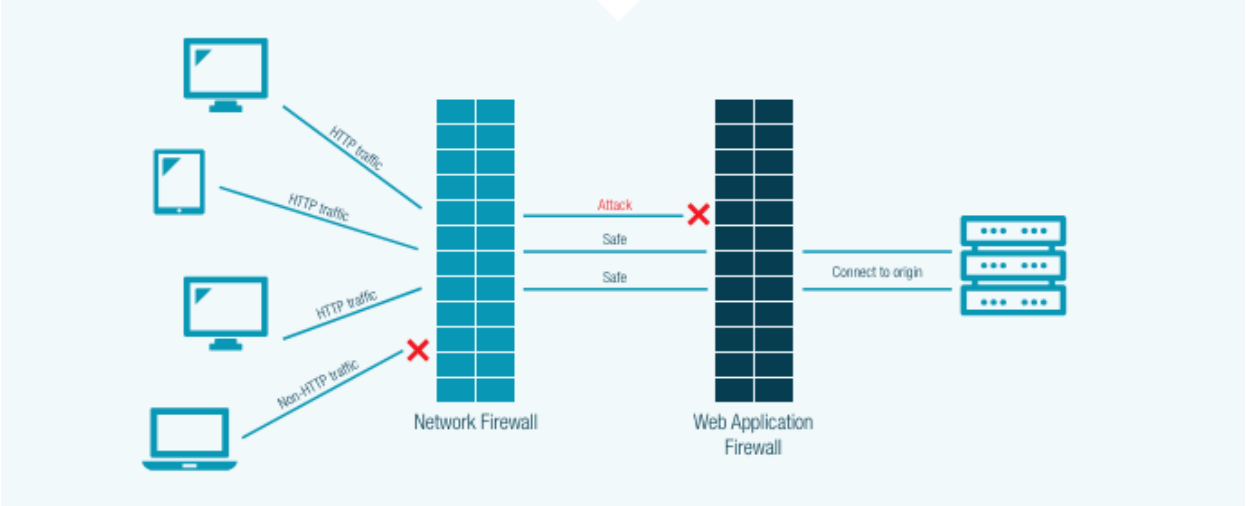


Рисунок 0.1 - Як працює WAF

В основному правила WAF застосовуються до початку найбільш загальних нападів так:

1. Міжсайтовий сценарій (XSS) – зловмисний HTML-код, вставлений хакером у поле введення веб-сторінки

2. Поверхістна поправка – хакери переписують вихідний код веб-сторінки, щоб змінити значення, що зберігаються в прихованих полях, а потім відправляють змінений код назад на сервер

3. Отруєння кукі – параметр зміни

4. Web scraping – автоматизоване вилучення даних з веб-сторінок

5. Рівень 7 DoS-атаки – переважний веб-сервер з використанням активного використання

6. Зміни параметрів – позначення значень у параметрах виклику веб-сторінки

7. Buffer over ow – введення користувача, яке перезаписує код у пам’яті

8. Backdoor або Debug options – повідомлення розробників для тестування веб-сторінок, які можуть бути використані як хакери для доступу до процесора

9. Stealth commanding – атака на операційну систему веб-сервера

10. Примусовий перегляд – хакер отримує доступ до резервних копій або тимчасових папок на веб-сервері

11. Помилки третіх сторін – маніпулювання вмістом, наданим іншими компаніями

12. SQL-ін’єкції – запити, введені в поля аутентифікації користувача

Хоча проксі зазвичай захищають клієнтів, WAF захищають сервери. WAF розгортається перед веб-серверами для захисту певної веб-програми або набору веб-програм від атак. WAF можна розглядати як зворотний проксі. WAF можуть бути у формі пристрою, плагіна сервера або фільтра та можуть бути налаштовані відповідно до програми. Зусилля для виконання цього налаштування можуть бути значними, і їх потрібно підтримувати під час модифікації програми. В епоху хмари WAF також еволюціонують до «служби в хмарі», що може означати дуже різні речі.

RASP (Runtime Application Self-Protection)

Технологія самозахисту програми під час виконання (RASP) визначає та блокує загрози безпеці програми в режимі реального часу. Додавши функції виявлення та захисту до середовища виконання додатків, RASP дозволяє додаткам

«самозахищатися» шляхом автоматичної зміни конфігурації, без втручання людини, у відповідь на певні умови.

У реальному часі RASP аналізує як поведінку програми, так і контекст поведінки. Таким чином, він реалізує постійний аналіз безпеки, при цьому система негайно реагує на будь-які розпізнані атаки. Ця контекстно-залежна можливість також дозволяє розгортати RASP з мінімальними попередніми налаштуваннями або поточним обслуговуванням; RASP розуміє, як дані використовуються в програмі для автоматичного впровадження захисту програми.

RASP стане важливим гравцем на ринку AppSec у майбутньому, але все ще є технологією на ранній стадії розвитку. Більшість компаній сьогодні покладаються на більш зрілі технології, такі як статичний аналіз, динамічний аналіз і аналіз складу програмного забезпечення, щоб захистити свої портфоліо програм. Але потрібно слідкувати за цією технологією в майбутньому, вона дуже перспективна.

Програми є основним вектором атак для кіберзлочинців. І оскільки організації все більше покладаються на веб-, мобільні та хмарні програми для розвитку свого бізнесу, кількість загроз різко збільшилась.

Немає єдиного чарівного рішення для захисту від усіх загроз програм. Підприємствам потрібне комплексне рішення для зменшення ризиків на кожному етапі життєвого циклу програми. Подібно до того, як підприємства розгортають кілька рівнів фізичної безпеки (двері, замки, значки, камери спостереження тощо), вони повинні застосовувати кілька рівнів кібербезпеки для захисту додатків, які керують їхнім бізнесом протягом усього життєвого циклу – від розробки, тестування до виробництва.

Агент RASP, який знаходиться в середовищі виконання, бачить перебіг програми в режимі реального часу та використовує контекстне розуміння для виявлення, перевірки та припинення атак у робочих програмах. Це детальне уявлення про дії системи, включаючи розуміння логіки програми, конфігурації та потоків даних і подій, покращує точність і мінімізує помилкові спрацьовування. У дизайні додатків.

Прикладом умови, яка може викликати відповідь RASP, є виконання інструкцій, які звертаються до бази даних (що може спричинити експлойт ін’єкції SQL). Технологія може перебувати в режимі діагностики та просто подавати звуковий сигнал щодо атаки, або вона може бути в режимі самозахисту та зупиняти потенційно зловмисне виконання.

RASP — це єдина технологія, яка може блокувати загрози точно та безперервно, з мінімальною участю людини та нульовою затримкою.

IDS/IPS системи (Intrusion Detection/Prevention Systems)

Виявлення вторгнень — це процес моніторингу подій, що відбуваються в комп’ютерній системі чи мережі, та аналізу їх на наявність ознак можливих інцидентів, які є порушеннями або неминучою загрозою порушення політик комп’ютерної безпеки, політик прийнятного використання або стандартних практик безпеки. Система виявлення вторгнень (IDS) — це програмне забезпечення, яке автоматизує процес виявлення вторгнень. Система запобігання вторгненням (IPS) — це програмне забезпечення, яке має всі можливості системи виявлення вторгнень і може також намагатися зупинити можливі інциденти. Технології IDS і IPS пропонують багато однакових можливостей, і адміністратори зазвичай можуть вимкнути функції запобігання в продуктах IPS, змушуючи їх функціонувати як IDS. Відповідно, для стислості термін системи виявлення та запобігання вторгненням (IDPS) використовується в решті цього розділу для позначення технологій IDS і IPS. Будь-які винятки окремо зазначаються.

IDPS в першу чергу зосереджені на виявленні можливих інцидентів. Наприклад, IDPS може виявити, коли зловмисник успішно скомпрометував систему, використовуючи вразливість у системі. IDPS буде реєструвати інформацію про діяльність і повідомляти про інцидент адміністраторам безпеки, щоб вони могли ініціювати дії реагування на інцидент, щоб мінімізувати шкоду. Багато IDPS також можуть бути налаштовані на розпізнавання порушень політики прийнятного використання та інших політик безпеки — наприклад, використання заборонених програм однорангового обміну файлами та передача великих файлів бази даних на знімні носії або мобільні пристрої. Крім того, багато IDPS можуть

ідентифікувати розвідувальну діяльність, яка може вказувати на неминучість атаки або на те, що певна система чи характеристика системи становить особливий інтерес для зловмисників. Інше використання IDPS полягає в тому, щоб краще зрозуміти загрози, які вони виявляють, зокрема частоту та характеристики атак, щоб можна було визначити відповідні заходи безпеки. Деякі IDPS також можуть змінювати свій профіль безпеки, коли виявляється нова загроза. Наприклад, IDPS може збирати більш детальну інформацію для певного сеансу після виявлення зловмисної активності в цьому сеансі.

Технології IPS відрізняються від технологій IDS однією характеристикою: технології IPS можуть реагувати на виявлену загрозу, намагаючись запобігти її успіху. Вони використовують кілька прийомів реагування, які можна розділити на такі групи:

- IPS сама зупиняє атаку. Приклади того, як це можна зробити, включають IPS, що розриває мережеве з'єднання, яке використовується для атаки, і IPS, що блокує доступ до цілі з облікового запису користувача, IP-адреси чи іншого атрибута зловмисника.
- IPS змінює середовище безпеки. IPS може змінити конфігурацію інших засобів безпеки, щоб перешкодити атаці. Типовими прикладами є перенастроювання IPS мережевого брандмауера для блокування доступу зловмисника або цілі, а IPS змінює брандмауер на основі 2 хостів на цілі для блокування вхідних атак. Деякі IPS можуть навіть спричинити застосування патчів до хосту, якщо IPS виявляє, що хост має вразливі місця.
- IPS змінює зміст атаки. Деякі технології IPS можуть видалити або замінити шкідливі частини атаки, щоб зробити її доброякісною. Простим прикладом є IPS, який видаляє заражений вкладений файл із електронного листа, а потім дозволяє очищеному електронному листу дістатися до одержувача. Більш складним прикладом є IPS, який діє як проксі та нормалізує вхідні запити, що означає, що проксі перепаковує корисні навантаження запитів, відкидаючи інформацію заголовка. Це може призвести до того, що певні атаки будуть відхилені в рамках процесу нормалізації.

2.2. Аналіз ефективності різних методів захисту

Порівняльний аналіз WAF та RASP

Брандмауер веб-додатків (WAF) зазвичай знаходиться перед веб-додатками, перевіряючи вхідний трафік HTTP-запитів на наявність відомого корисного навантаження атак і ненормальних моделей використання. Коли виявлено підозріле корисне навантаження або використання, WAF може повідомити про порушення або повідомити та заблокувати запит. WAF є чудовим варіантом для захисту від наступних типів атак: об’ємної (DDoS), автоматизації ботів (збирання, сканування тощо), захоплення облікового запису (внесення облікових даних) і безпеки API (північ/південь).

RASP пропонує критичний рівень запобігання атак за WAF або наступного покоління WAF. Підхід RASP відрізняється від традиційних WAF тим, що він тісно пов’язаний із кодом програми. RASP використовує контекстну обізнаність, без чорних або білих списків, щоб виявити загрози та забезпечити впевненість, що конкретне корисне навантаження не зможе використати невідому частину коду програми. Технологія RASP перевіряє повні (і часто перетворені дані) у контексті того, як програма використовуватиме їх, якщо і тільки якщо програма намагатиметься використати дані. Результатом є низька кількість помилкових спрацьовувань і висока видимість вразливостей, включаючи слабкі сторони, раніше невідомі організації. Таким чином, RASP доповнює WAF і служить останньою лінією захисту в безпеці програм(рисунок 2.2).

Criteria	Deloitte's Runtime Application Self Protection (RASP) Service	Web Application Firewall (WAF) Deployments
Accuracy	Detection of malicious input only when passed to library calls where exploitation would occur. Monitors inbound and outbound data and logic flows	Detection is based on naïve pattern matching, without considering whether the input data would be passed to vulnerable code
Time to Value	No need to know locations of existing vulnerabilities in application code; can act as a virtual patch against a vulnerability	Requires extensive testing and configration to adequately cover the application. It also involves fine tuning
Reliability	Will not fail open under high load–code is always instrumented, regardless of servers load	Single point of failure; likely to fail open under high load, leaving the web application vulnerable
Platforms	Any instrumented application	All types of Web applicattion
Visibility	Provides detailed feedback to developers to show how to remediate code vulnerabilities	Offers no detailed insight into the application
Network Protocols	Protocol agnostic; handles HTTP, HTTPS, AJAX, SQL and SOAP with equal ease	Must be able to understad the application's netwotk language type
Maintenance	Automatically understands changes to the application	Can gain application context through training only

Рисунок 0.2 - порівняння RASP і WAF

Використання штучного інтелекту в системах IDS/IPS

Виявлення та запобігання загрозам:

- Системи штучного інтелекту аналізують великі набори даних у реальному часі, включаючи відео, чат, електронні листи тощо, щоб виявити шаблони та аномалії, що вказують на потенційні загрози безпеці. Припустімо, що кіберзлочинці на крок попереду в маніпулюванні кодом. У такому сценарії машинне навчання може вчитися на їхніх історичних даних і виявляти нові варіанти, навіть якщо шкідливі загрози ховаються на виду в коді.
- Сканування історичних даних і відомостей про поточні загрози допомагає компаніям проводити прогнозний аналіз, допомагаючи їм визначати критичні вразливості, визначати пріоритети, які з них потрібно усунути в першу чергу, і оптимізувати розподіл ресурсів там, де це найбільше потрібно. Наразі 63% порушень можна виявити за лічені хвилини за допомогою штучного інтелекту.

Аналіз поведінки:

- На відміну від традиційної кібербезпеки, яка значною мірою покладається на виявлення на основі сигнатур, аналіз поведінки за допомогою ШІ охоплює дії користувачів, мережевий трафік і системні події. Він вивчає та встановлює базову лінію для «нормальної» поведінки, специфічної для середовища, роблячи цю базову лінію контрольною точкою для виявлення аномалій.
- Постійно відстежуючи відхилення від встановлених норм, штучний інтелект у сфері кібербезпеки може виявляти незвичайні спроби входу, раптові сплески доступу до даних, несанкціонований доступ до файлів або системи та відхилення від типових шаблонів зв’язку. Це дає можливість підприємствам раннього виявлення, швидшого реагування за допомогою автоматизованих сповіщень або процедур стримування, що зрештою мінімізує потенційну шкоду.

Автоматизоване реагування на інциденти:

AI автоматизує реагування на інциденти, швидко виявляючи та стримуючи інциденти безпеки, щоб мінімізувати час відповіді та вплив. Ці системи адаптуються та навчаються на кожному інциденті, підвищуючи загальну безпеку та звільняючи спеціалістів із безпеки, щоб зосередитися на стратегічних ініціативах і складних розслідуваннях. Це спрощує процес і зменшує операційне навантаження, що дозволяє організаціям завчасно зміцнювати свою позицію щодо кібербезпеки.

Виявлення фішингу та шахрайства:

AI покращує виявлення фішингу та шахрайства за допомогою NLP для фішингових електронних листів і машинного навчання для виявлення незвичайних моделей транзакцій або поведінки користувачів. Інструменти виявлення фішингу на основі штучного інтелекту допомагають фахівцям із безпеки відфільтровувати помилкові спрацьовування, що дозволяє їм зосередитися на справжніх загрозах і мінімізує ризик не помітити критичні сповіщення безпеки.

Краща безпека мережі:

- Системи виявлення та запобігання вторгнень (IDPS) на основі штучного інтелекту: IDPS на основі штучного інтелекту може виявляти мережеві загрози та реагувати на них у реальному часі, захищаючи мережі від несанкціонованого доступу та атак.
- Покращення брандмауера: штучний інтелект може оптимізувати правила брандмауера на основі моделей мережевого трафіку, підвищуючи ефективність безпеки мережі
- SIEM (Інформація про безпеку та керування подіями): штучний інтелект може покращити аналіз подій безпеки, зіставляючи дані з різних джерел для виявлення складних загроз.

Оцінка проникнення (Penetration Testing)

Тестування на проникнення(рисунок 2.3), тестування пера або етичне хакерство – це процес оцінки програми чи інфраструктури на наявність уразливостей у спробі використати ці вразливості та обійти або зруйнувати

функції безпеки компонентів системи за допомогою ретельного ручного тестування. Ці вразливості можуть існувати через неправильну конфігурацію, незахищений код, погано розроблену архітектуру або розкриття конфіденційної інформації серед інших причин. Результатом є дієвий звіт, у якому пояснюється кожна вразливість або ланцюжок уразливостей, використаних для отримання доступу до цілі, із зазначенням кроків, вжитих для їх використання, а також докладна інформація про те, як їх виправити, і додаткові рекомендації. Кожній виявленій вразливості присвоюється рейтинг ризику, який можна використовувати для визначення пріоритетності ефективних завдань усунення.

1. Test preparation.
2. Test implementation.
3. Test analysis.

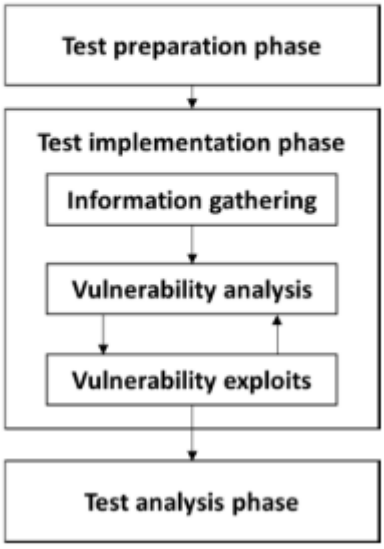


Рисунок 0.3 - Алгоритм тесту на проникнення

Тест на проникнення використовується для визначення ризиків, які можуть виникнути, коли зловмисник отримує доступ до обчислювальної системи та мереж організації. Виконання тесту PEN допоможе оцінити план пом’якшення, щоб усунути прогалини в безпеці до того, як станеться фактична атака. Проведення PEN-тесту допомагає організаціям зменшити фінансові та інформаційні втрати, які спричинили б втрату довіри клієнтів через порушення безпеки. Він захищає організації від збоїв шляхом запобігання фінансовим

втратам і забезпечує відповідність галузевим регуляторам, клієнтам і акціонерам; сприяння розвитку довіри, корпоративного іміджу та раціоналізації інвестицій у безпеку IT. Оскільки тестування на проникнення є проактивним процесом, воно надає незаперечну інформацію, яка допомагає організації відповідати аспектам аудиту або дотримання нормативних вимог. Однією з головних цілей PEN-тестування є створення IT-безпеки та її важливість на всіх рівнях. організації за допомогою структурованих програм навчання та підвищення обізнаності, щоб уникнути інцидентів безпеки, які можуть завдати шкоди конфіденційності, цілісності, відносинам і довірі клієнтів. PEN-тест допомагає організації оцінити рівень обізнаності про безпеку серед її співробітників, ефективність існуючої політики та процесу безпеки, а також ефективність її продуктів. Це допомагає в процесі прийняття рішень для оцінки безпеки організації та, отже, планування інвестицій у безпеку та IT-стратегії. Тестування на проникнення також допомагає у формуванні важливих аспектів стратегії інформаційної безпеки шляхом швидкого й точного виявлення вразливостей. Він також підтримує вдосконалення тестових конфігурацій для проактивного усунення виявлених ризиків. Це допомагає бізнесу оцінити наслідки та ймовірність уразливості. Таким чином, організація може визначити пріоритети та реалізувати план дій щодо пом’якшення виявлених вразливостей. Тестування на проникнення забирає багато часу, зусиль і знань в залежності від складності бізнесу. Таким чином, тестування на проникнення сприяє підвищенню знань і компетентності осіб, залучених до процесу. Його вважають інструментом забезпечення якості, який приносить користь як бізнесу, так і операційній діяльності.

Комплексні рішення з кібербезпеки (Cisco)

Огляд безпеки Cisco

Безпека Cisco — це багатогранний підхід, який охоплює як апаратне, так і програмне забезпечення для досягнення повної безпеки мережі. Широкий асортимент апаратних і програмних продуктів включає продукти безпеки, призначені для:

- Управління доступом
- Розширений захист від шкідливих програм
- Безпека електронної пошти
- Брандмауери нового покоління
- Системи запобігання вторгненням нового покоління
- Безпека маршрутизатора та комутатора
- Управління безпекою
- Клієнти безпеки VPN
- Веб-безпека

Звичайно, кожен із них є предметом глибокого обговорення, тому, якщо ви не знайомі з їхнім набором продуктів, ми рекомендуємо зателефонувати нам і поговорити про те, що є правильним для безпеки вашої мережі.

У перерахованих вище сферах, як-от безпека електронної пошти та захист від зловмисного програмного забезпечення, вони пропонують різні лінії продуктів для протидії можливим атакам на кожному фронті:

Розширений захист від загроз

Їхній набір засобів захисту від загроз розроблений для превентивного аналізу потенційних атак, захисту та відповідного реагування. Цей набір продуктів включає Advanced Malware Protect (AMP), AMP ThreatGrid, Cognitive Threat Analysis і Cyber Threat Defense. Це просунуте проактивне рішення є важливим у галузі та призначене для компаній, які прагнуть бути в авангарді мережевої безпеки.

Безпека мережі та центру обробки даних

Незважаючи на те, що безпека мережі отримує найбільшу славу та захист у світі кібербезпеки, багато хакерів прагнуть не вашої мережі, а даних у ній. Щоб захистити ваші дані, які зберігаються на сайті або за його межами (але не обов’язково є частиною ваших щоденних процесів застосування), вам знадобляться продукти безпеки мережі та центру обробки даних Cisco. До них належать:

- Брандмауери нового покоління
- Розширений захист від шкідливих програм
- Запобігання вторгненням нового покоління
- Програмно-визначена сегментація Cisco TrustSec
- Безпечний доступ
- Захист від кіберзагроз
- Безпечний центр обробки даних
- Інфраструктура, орієнтована на застосування
- Швидке стримування загрози

Використовуючи продукти, перелічені вище, ви можете успішно підвищити безпеку свого центру обробки даних за допомогою таких елементів, як безпечна віртуалізація, сегментація даних, а також керування користувачами та іншими правилами.

Мобільна безпека та рішення для кінцевих точок

Мобільна безпека є основною проблемою для багатьох компаній, особливо для тих, які працюють під егідою BYOD або політики для кількох пристроїв. Рішення Cisco для захисту мобільних пристроїв і кінцевих точок використовують контекстно-залежну безпеку, яка розширює можливості ваших користувачів, одночасно захищаючи вашу мережу. Цей набір продуктів включає їх Cisco AnyConnect Secure Mobility, Cisco Secure Access і їх Advanced Protection Malware.

Безпека Інтернету, електронної пошти та Cloud Gateway

Основна частина загроз безпеці походить від перегляду веб-сторінок, електронної пошти та інших онлайн-платформ, де на допомогу приходить безпека шлюзу Cisco. Це поєднання апаратного та програмного забезпечення використовує різноманітні продукти, зокрема: Web Security, Cloud Web Security, Cisco Cloud Access Безпека та Безпека електронної пошти. Поєднання всіх цих продуктів захищає безпеку вашої мережі на передньому краї загроз безпеки в Інтернеті.

2.3. Найвідоміші кібератаки у вебі

SQLi Атака на Sony Pictures

LulzSec, хакерська група, яка вже потрапляла в новини щодо злому PBS, 2 липня 2011 року заявила, що зламала кілька веб-сайтів Sony Pictures і отримала доступ до незашифрованої особистої інформації понад 1 мільйона людей.

У заяві, опублікованій у той же четвер, група стверджувала, що їй також вдалося скомпрометувати всі «реквізити адміністратора», включаючи паролі адміністратора, а також 75 000 «музичних кодів» і 3,5 мільйона «музичних купонів» з мереж і веб-сайтів Sony.

Група публічно опублікувала повний список скомпрометованих сайтів разом із посиланнями на документи, що містять зразки того, що, як вона стверджувала, було викраденим у Sony.

Зламані бази даних включали одну, яка містила інформацію, що належить людям, які брали участь у рекламній кампанії за участю Sony Pictures і AutoTrader.com, а також іншу, яка включала спонсоровану Sony кампанію Summer of Restless Beauty.

За даними LulzSec, під час злому також було зламано базу даних музичних кодів Sony, базу даних музичних купонів і бази даних Sony BMG Belgium & Netherlands.

За словами групи, скомпрометовані бази даних містили «різноманітну інформацію про користувачів і співробітників Sony».

«SonyPictures.com володів дуже простою SQL-ін’єкцією, однією з найпримітивніших і найпоширеніших вразливостей, як ми всі вже повинні знати», — сказав LulzSec. «Від однієї ін’єкції ми отримали доступ до ВСЬОГО».

«Що гірше те, що кожен біт даних, який ми взяли, не був зашифрований», — стверджує група. «Sony зберігала понад 1 000 000 паролів своїх клієнтів у відкритому вигляді, а це означає, що їх просто взяти».

LulzSec заявив, що скопіював і опублікував лише відносно невелику частину інформації, до якої вдалося отримати доступ, оскільки у нього не було ресурсів, щоб завантажити все. Група заявила, що теоретично вона могла б «взяти всю інформацію до останнього», але це зайняло б тижні.

XSS атака на Твіттер

Згідно з AFP, фестиваль вірусів у Twitter розпочався з 17-річного австралійця на ім'я Пірс Делфін.

Підліток виявив проблему, яка дозволяла з'являтися коду javascript у твітах. Він опублікував деякий код, який згодом був підхоплений хакерами та використаний для більш зловмисних цілей, включаючи різноманітні зашифровані повідомлення у вікні та перенаправлення на японський порносайт. Вірус поширювався легко, оскільки для його активації користувачам достатньо було лише навести курсор на посилання, а не клацати по ньому.

"Я зробив це, щоб побачити, чи це можливо... що JavaScript дійсно може бути виконаним у твіті", - розповів Делфін AFP. "У момент публікації твіта я не уявляв, що він набере такої популярності. Я навіть не розглядав такий варіант."

Twitter перебував у хаосі приблизно п'ять годин, поки помилку не було виправлено. New York Times повідомила, що Twitter знав про проблему ще в серпні та виправив її, хоча оновлення, не пов'язане з оновленням дизайну минулого тижня, відновило проблему.

Twitter пояснив: "Сьогодні вранці користувач помітив проблему з безпекою та скористався нею на Twitter.com. Спочатку хтось створив обліковий запис, який використовував цю проблему, змінюючи кольори твітів і викликаючи вікно з текстом, коли хтось наводив курсор на посилання у твіті. Ось чому люди називають це помилкою 'onMouseOver' – експлуатація відбувалася, коли хтось наводив курсор на посилання."

Інші користувачі пішли далі та додали код, який змушував людей ретвітити оригінальний твіт без їхнього відома.

Серед постраждалих були прес-секретар Білого Дому Роберт Гіббс і Сара Браун, дружина колишнього прем'єр-міністра Гордона. Делфін також зазначив, що якби хакери могли впоратися з викликом у 140 символів, вони могли б використовувати цю проблему для витягу інформації про паролі.

Висновки

У другому розділі ми детально розглянули існуючі рішення у сфері кібербезпеки, зосередившись на популярних інструментах захисту веб-додатків, проаналізувавши ефективність методів захисту від різноманітних і найвідоміших кібератак у Мережі. Ми почали з аналізу популярних засобів захисту веб-додатків, таких як брандмауери веб-додатків (WAF), самозахист додатків під час виконання (RASP) і системи запобігання/виявлення вторгнень (IDS/IPS). Кожен із цих інструментів було детально розглянуто, включаючи функціональні можливості, принципи роботи та приклади використання в контексті реального світу. Це дозволило нам зрозуміти, як інтегрувати ці технології в системи для забезпечення надійного захисту.

Далі проводили порівняльний аналіз ефективності різних методів захисту. Зокрема, ми порівнювали WAF і RASP, розглядаючи їхні переваги та недоліки в різних контекстах використання. Ми також дослідили роль штучного інтелекту в системах IDS/IPS, який може значно підвищити ефективність виявлення та запобігання кібератакам.

Нарешті, ми розглянули комплексні рішення кібербезпеки, запропоновані Cisco, які поєднують багато різних інструментів і методів захисту в одній системі. На завершення ми проаналізували найвідоміші кібератаки в Інтернеті, такі як атака SQLi на Sony Pictures і атака XSS на Twitter. Вивчення цих випадків дозволяє нам зрозуміти реальні загрози та наслідки кібератак, а також важливість впровадження ефективних заходів захисту.

РОЗРОБКА НАДІЙНОГО ВЕБ-ЗАСТОСУНКУ

Якісний веб-застосунок складається з багатьох факторів:

1. Функціональність: Веб-застосунок повинен виконувати всі необхідні функції та відповідати вимогам користувача.
2. Зручність використання (юзабіліті): Інтерфейс повинен бути інтуїтивно зрозумілим, простим у навігації та зручним для користувача.
3. Продуктивність: Застосунок повинен швидко завантажуватись і реагувати на дії користувача.
4. Безпека: Застосунок повинен мати надійні засоби захисту від несанкціонованого доступу та зловмисних атак.
5. Масштабованість: Застосунок повинен мати можливість легко адаптуватись до зростаючого числа користувачів та об'єму даних.
6. Сумісність: Застосунок повинен коректно працювати на різних платформах, браузерах та пристроях.
7. Надійність: Застосунок повинен мати мінімальний час простою і бути стійким до збоїв.
8. Модульність: Архітектура застосунку повинна бути модульною, що дозволяє легко додавати нові функції та оновлення.
9. Зручність обслуговування: Застосунок повинен мати прості механізми для оновлення, виправлення помилок та технічної підтримки.
10. Доступність: Застосунок повинен бути доступним для користувачів з обмеженими можливостями, зокрема забезпечувати підтримку для екранних рідерів та інших допоміжних технологій.

Оскільки більшість з цих опцій пов’язані одна з одною ми реалізуємо їх всіх у одному веб-еплікейшні типу «Цифровий гаманець» з яким я працював підчас переддипломної практики у компанії «Епам Діджитал»

3.1 З’ясування проблеми та встановлення обсягу роботи

Давайте виділимо функціональні і нефункціональні вимоги:

1. Підтримка операції переказу балансу між двома цифровими гаманцями(рисунок 3.1).
2. Підтримка 1 000 000 TPS(транзакцій за секунду).
3. Надійність не менше 99,99%.
4. Підтримка транзакцій.
5. Підтримка відтворюваності.

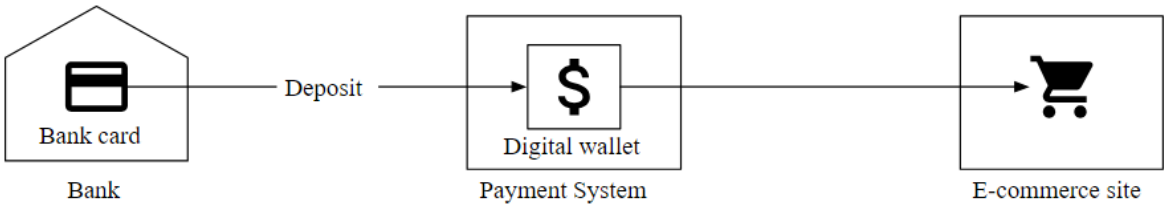


Рисунок 0.1 - схема роботи веб гаманця

Коли ми говоримо про TPS, ми маємо на увазі використання транзакційної бази даних. Сьогодні реляційна база даних, що працює на типовому вузлі центру обробки даних, може підтримувати кілька тисяч транзакцій на секунду. Припустімо, що вузол бази даних може підтримувати 1000 TPS. Щоб досягти 1 мільйона TPS, нам потрібно 1000 вузлів бази даних.

Однак цей розрахунок трохи неточний. Кожна команда переказу вимагає двох операцій: зняття грошей з одного рахунку та внесення грошей на інший рахунок. Щоб підтримувати 1 мільйон передач за секунду, система фактично повинна обробляти до 2 мільйонів TPS, що означає, що нам потрібно 2000 вузлів.

У таблиці 3.1 показано загальну кількість вузлів, необхідних, коли змінюється «TPS для кожного вузла» (TPS, який може обробляти один вузол). Якщо припустити, що апаратне забезпечення залишається незмінним, чим більше транзакцій один вузол може обробити за секунду, тим менша загальна кількість вузлів потрібна, що вказує на нижчу вартість обладнання. Таким чином, одна з наших цілей розробки полягає в тому, щоб збільшити кількість транзакцій, які може обробити один вузол.

TPS для кожного вузла	Число вузлів
100	20000
1000	2000
10000	200

Таблиця 0.1 - кількість вузлів

3.2 Розробляємо поверхневий дизайн системи

Дизайн АПІ

Ми будемо використовувати конвенцію RESTful API. Щоб зменшити обсяг програми для даної роботи ми будемо підтримувати лише один API(таблиця 3.2):

АПІ	Деталі
POST /v1/wallet/balance_transfer	Перенести баланс з одного гаманця у інший

Таблиця 0.2 - ендпоінт

Параметри запиту(таблиця 3.3):

Поле	Опис	Тип
from_account	Дебетовий рахунок	string
to_account	Кредитний рахунок	string
amount	Сума грошей	string
currency	Тип валюти	string (ISO 4217)
transaction_id	Айді	uuid

Таблиця 0.3 - параметри запиту

Зразок тіла відповіді(рисунок 3.2):

```
{
"Status": "success"
"Transaction_id": "01589980-2664-11ec-9621-0242ac130002"
}
```

Рисунок 0.2 - тіло відповіді

Рішення для сегментування в пам’яті

Програма гаманця підтримує баланс кожного облікового запису користувача. Хорошою структурою даних для представлення цього відношення <user,balance> є карта, яка також називається хеш-таблицею (картою) або сховищем ключ-значення.

Для сховищ у пам’яті популярним вибором є Redis. Одного вузла Redis недостатньо для обробки 1 мільйона TPS. Нам потрібно налаштувати кластер вузлів Redis і рівномірно розподілити між ними облікові записи користувачів. Цей процес називається розділенням або шардингом.

Щоб розподілити дані про ключ-значення між N розділами, ми можемо обчислити хеш-значення ключа та розділити його на N. Залишок є місцем призначення розділу. Псевдокод нижче показує процес шардингу(рисунок 3.3):

```
String accountID = "A";
Int partitionNumber = 7;
Int myPartition = accountID.hashCode() % partitionNumber;
```

Рисунок 0.3 - код шардингу

Як зробити оновлення для двох різних вузлів зберігання атомарними? Першим кроком є заміна кожного вузла Redis на транзакційний вузол реляційної бази даних. На рисунку 3.4 показана архітектура. Розділимо клієнтів A, B і C на 3 реляційні бази даних, а не на 3 вузли Redis.

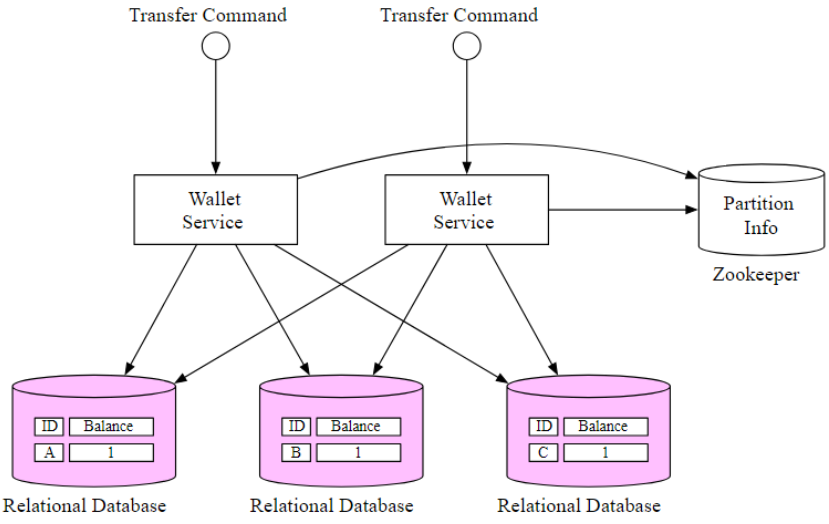


Рисунок 0.4 - архітектура з реляційними бд

Використання транзакційних баз даних вирішує лише частину проблеми. Як згадувалося в останньому розділі, дуже ймовірно, що одна команда передачі потребуватиме оновлення двох облікових записів у двох різних базах даних. Немає гарантії, що дві операції оновлення буде виконано одночасно. Якщо служба гаманця перезапустилася відразу після оновлення балансу першого облікового запису, як ми можемо переконатися, що другий обліковий запис також буде оновлено?

Розподілена транзакція: двофазова фіксація

У розподіленій системі транзакція може включати кілька процесів на кількох вузлах. Щоб зробити транзакцію атомарною, відповіддю може бути розподілена транзакція. Існує два способи реалізації розподіленої транзакції: низькорівневе рішення та високорівневе рішення. Ми розглянемо низькорівневе.

Рішення низького рівня покладається на саму базу даних. Найпоширеніший алгоритм називається двофазовим фіксуванням (2PC). Як випливає з назви, він має дві фази, як на рисунку 3.5.

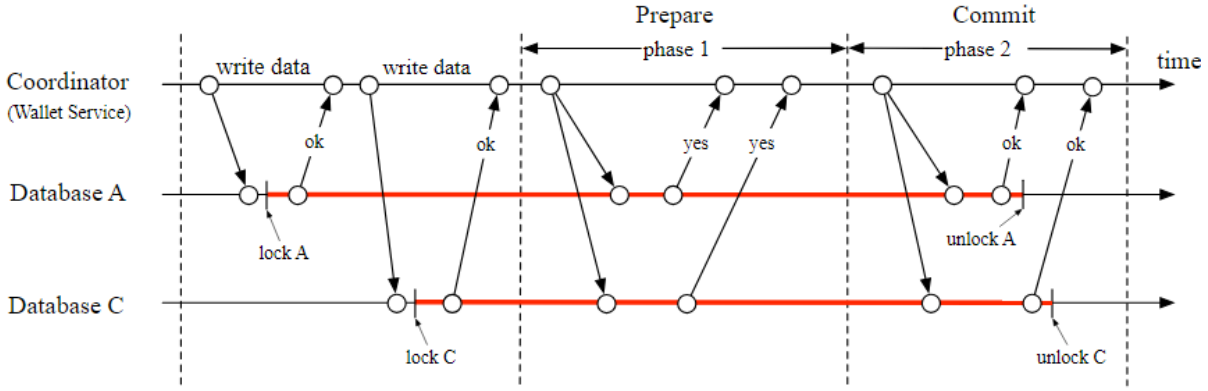


Рисунок 0.5 - двофазне фіксування

1. Координатор, яким у нашому випадку є служба гаманця, виконує операції читання та запису в кількох базах даних як зазвичай. Як показано на малюнку 5, обидві бази даних A і C заблоковані.

2. Коли програма збирається зафіксувати транзакцію, координатор просить усі бази даних підготувати транзакцію.

3. На другому етапі координатор збирає відповіді з усіх баз даних і виконує наступне:

a. Якщо всі бази даних відповідають «так», координатор просить усі бази даних зафіксувати транзакцію, яку вони отримали.

b. Якщо будь-яка база даних відповідає «ні», координатор просить усі бази даних припинити транзакцію.

Це низькорівневе рішення, оскільки етап підготовки вимагає спеціальної модифікації транзакції бази даних. Наприклад, існує стандарт X/Open XA, який координує різнорідні бази даних для досягнення 2PC. Найбільша проблема 2PC полягає в тому, що він непродуктивний, оскільки блокування можна утримувати дуже довго, очікуючи повідомлення від інших вузлів. Ще одна проблема з 2PC полягає в тому, що координатор може бути єдиною точкою відмови, як показано на рисунку 3.6.

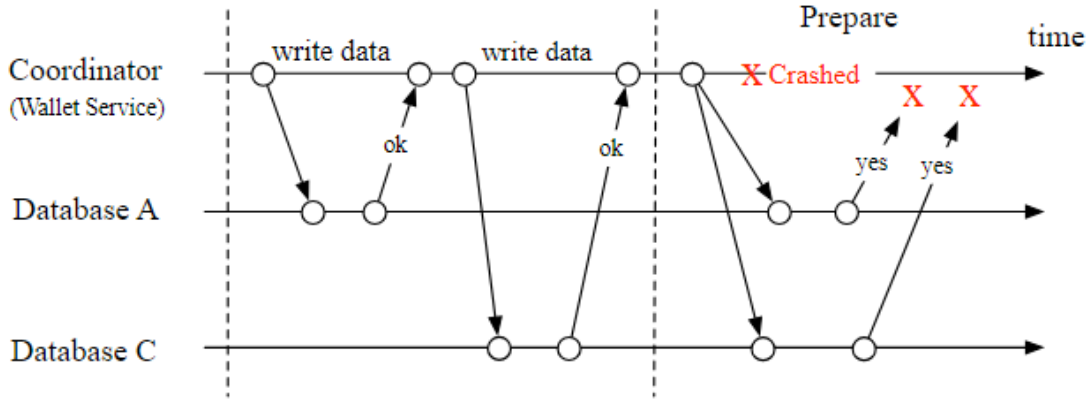


Рисунок 0.6 - 2PC

Event Sourcing

У реальному житті постачальник цифрового гаманця може бути перевірений. Ці зовнішні аудитори можуть поставити деякі складні запитання, наприклад:

1. Чи знаємо ми баланс рахунку в будь-який момент часу?
2. Як ми знаємо, що історичні та поточні баланси рахунку правильні?
3. Як довести, що системна логіка правильна після зміни коду?

Однією з філософій дизайну, яка систематично відповідає на ці запитання, є пошук подій, який є технікою, розробленою в Domain-Driven Design (DDD).

Є чотири важливі терміни в пошуку подій:

Команда

Команда — це передбачувана дія із зовнішнього світу. Наприклад, якщо ми хочемо переказати $1 від клієнта A до клієнта C, цей запит на переказ грошей є командою.

У пошуку подій дуже важливо, щоб усе було в порядку. Тому команди зазвичай поміщаються в чергу FIFO (першим увійшов, першим вийшов).

Подія

Команда – це намір, а не факт, оскільки деякі команди можуть бути недійсними та не можуть бути виконані. Наприклад, операція переказу буде невдалою, якщо після переказу баланс рахунку стане від’ємним.

Команда повинна бути перевірена, перш ніж ми щось з нею зробимо. Коли команда проходить перевірку, вона є дійсною та має бути виконана. Результат виконання називається подією.

Є дві основні відмінності між командою та подією.

1. Події повинні бути виконані, оскільки вони представляють підтверджений факт. На практиці ми зазвичай використовуємо минулий час для позначення події.
2. Команди можуть містити випадковість або введення/виведення, але події мають бути детермінованими. Події відображають історичні факти.

Є дві важливі властивості процесу генерації подій.

1. Одна команда може генерувати будь-яку кількість подій. Він може генерувати нуль або більше подій.
2. Генерація подій може містити випадковість, тобто не гарантується, що команда завжди генеруватиме однакові події. Генерація подій може містити зовнішні введення/виведення або випадкові числа. Ми ще раз розглянемо цю властивість більш детально в кінці розділу.

Порядок подій повинен відповідати порядку команд. Таким чином, події також зберігаються в черзі FIFO.

Стан

Стан – це те, що буде змінено під час виконання події. У системі гаманця стан – це баланси всіх рахунків клієнтів, які можна представити за допомогою структури даних карти. Ключ – це ім’я або ідентифікатор рахунку, а значення – баланс рахунку. Сховища ключ-значення зазвичай використовуються для зберігання структури даних карти. Реляційну базу даних також можна розглядати як сховище ключ-значення, де ключі є первинними ключами, а значення – рядками таблиці.

Стейт-менеджер

Стейт-менеджер керує процесом пошуку подій. Він виконує дві основні функції.

1. Перевіряє команди та генерує події.
2. Застосовує подію для оновлення стану.

Вибір джерела подій вимагає, щоб поведінка стейт-менеджера була детермінованою. Тому сам стейт-менеджер ніколи не повинен містити жодної випадковості. Наприклад, він ніколи не повинен читати нічого випадкового ззовні за допомогою вводу-виводу або використовувати будь-які випадкові числа. Коли він застосовує подію до стану, він завжди має генерувати той самий результат.

На малюнку 3.7 показано статичний вигляд архітектури джерела подій. Менеджер стану відповідає за перетворення команди на подію та за застосування події. Оскільки він має дві основні функції, ми зазвичай малюємо два стейт-менеджера один для перевірки команд, а інший для застосування подій.

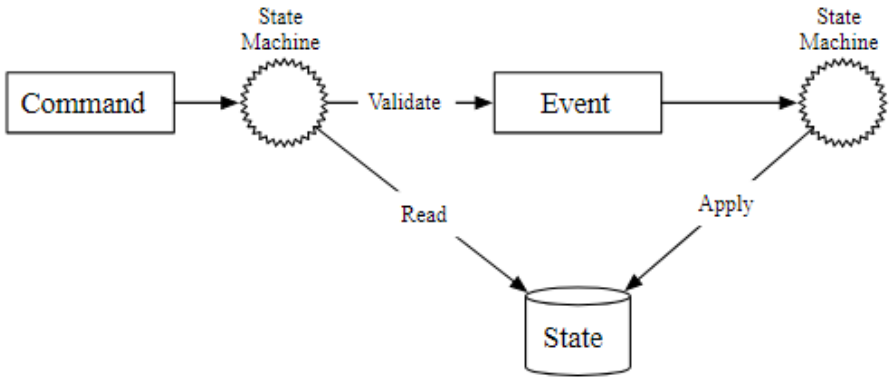


Рисунок 0.7 - джерело подій

Приклад сервісу гаманця

Для служби гаманця команди є запитами на переказ балансу. Ці команди поміщаються в чергу FIFO. Одним із популярних варіантів для черги команд є Kafka. Черга команд показана на малюнку 3.8.

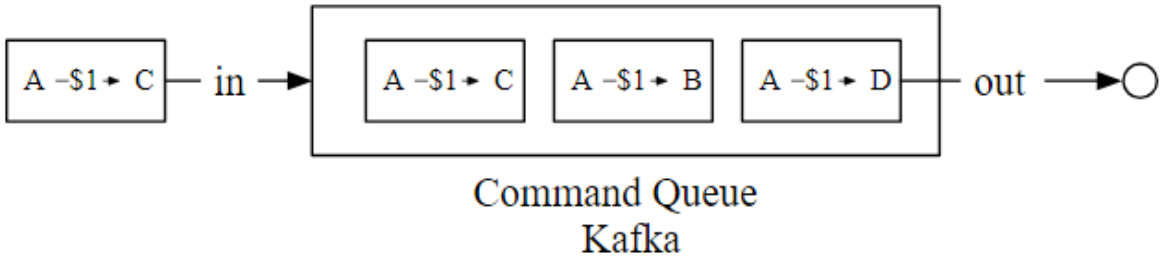


Рисунок 0.8 - черга команд

Припустімо, що стан (баланс рахунку) зберігається в реляційній базі даних. Кінцевий автомат перевіряє кожну команду одну за одною в порядку FIFO. Для кожної команди перевіряється, чи є на рахунку достатній баланс. Якщо так, кінцевий автомат генерує подію для кожного облікового запису.

На малюнку 3.9 показано, як працює стейт-менеджер у 5 кроків.

1. Читання команд із черги команд.
2. Зчитувати стан балансу з бази даних.
3. Перевірте команду. Якщо він дійсний, згенеруйте дві події для кожного з облікових записів.
4. Читайте наступну подію.
5. Застосуйте Подію, оновивши баланс в базі даних.

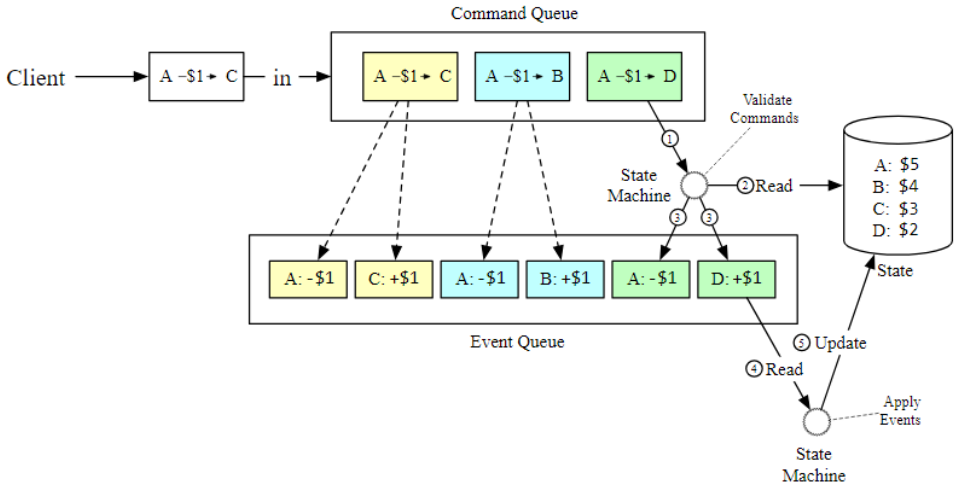


Рисунок 0.9 - менеджер стану

Відтворюваність

Найважливішою перевагою джерела подій перед іншими архітектурами є відтворюваність.

У рішеннях розподілених транзакцій, згаданих раніше, служба гаманця зберігає оновлений баланс рахунку (стан) у базі даних. Чому було змінено баланс рахунку, зрозуміти важко. Тим часом хронологічна інформація про баланс втрачається під час операції оновлення. У дизайні пошуку подій усі зміни спочатку зберігаються як незмінна історія. База даних використовується лише як оновлене уявлення про те, як виглядає баланс у певний момент часу.

Ми завжди можемо реконструювати історичні стани балансу, відтворюючи події з самого початку. Оскільки список подій є незмінним, а логіка кінцевого автомата є детермінованою, гарантується, що історичні стани, створені з кожного повтору, однакові.

На рисунку 3.10 показано, як відтворити стани служби гаманця шляхом відтворення подій.

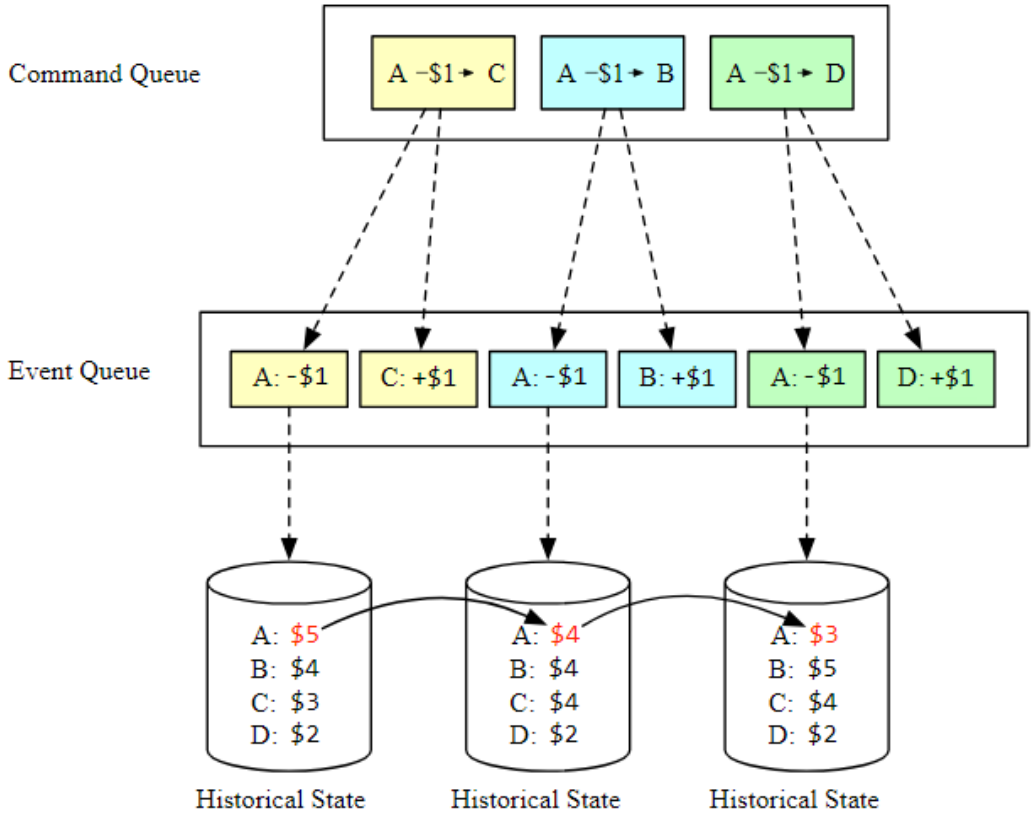


Рисунок 0.10 - відтворення подій у гаманці

Відтворюваність допомагає нам відповісти на складні запитання, які були названі вище:

1. Чи знаємо ми баланс рахунку в будь-який момент часу?
2. Як ми знаємо, що історичні та поточні баланси рахунку правильні?
3. Як ми доводимо, що логіка системи правильна після зміни коду?

На перше запитання ми могли б відповісти, відтворивши події від початку до моменту часу, коли ми хотіли б знати баланс рахунку.

Для другого запитання ми могли б перевірити правильність балансу рахунку, перерахувавши його зі списку подій.

Для третього запитання ми можемо запустити різні версії коду для подій і перевірити, що їхні результати ідентичні.

Через можливість аудиту джерело подій часто вибирається як фактичне рішення для служби гаманця.

CQRS

Наразі ми розробили службу гаманця для ефективного переміщення грошей з одного рахунку на інший. Однак клієнт досі не знає, який залишок на рахунку. Потрібен спосіб оприлюднити стан (інформацію про баланс), щоб клієнт, який перебуває за межами системи пошуку подій, міг знати, який він.

Інтуїтивно зрозуміло, ми можемо створити доступну лише для читання копію бази даних (історичний стан) і поділитися нею із зовнішнім світом. Джерело подій відповідає на це питання дещо по-іншому.

Замість того, щоб публікувати стан (інформацію про баланс), джерело подій публікує всі події. Зовнішній світ міг би сам зробити і кастомізувати під себе потрібний йому стейт. Ця філософія дизайну називається CQRS.

У CQRS є один стейт-менеджер, відповідальний за частину стану запису, але може бути багато менеджерів стану, які відповідають за формування переглядів станів. Ці перегляди можна використовувати для запитів.

Менеджери стану, зроблені лише для читання, можуть отримувати різні представлення стану з черги подій. Наприклад, клієнти можуть захотіти знати свої баланси, а стейт-менеджер може зберегти стан у базі даних для обслуговування

запиту балансу. Інший менеджер може створити стан для певного періоду часу, щоб допомогти дослідити такі проблеми, як можливі подвійні витрати. Інформація про стан – це слід для аудиторів, які можуть допомогти звірити фінансові записи.

Менеджери стейту, налаштовані тільки на читання, інколи мають затримки але завжди матимуть правильний результат. Цей архітектурний дизайн дозволяє нашій системі в кінці кінців завжди мати валідний результат, що є найважливішим у гаманцях.

На рисунку 3.11 показана класична архітектура CQRS.

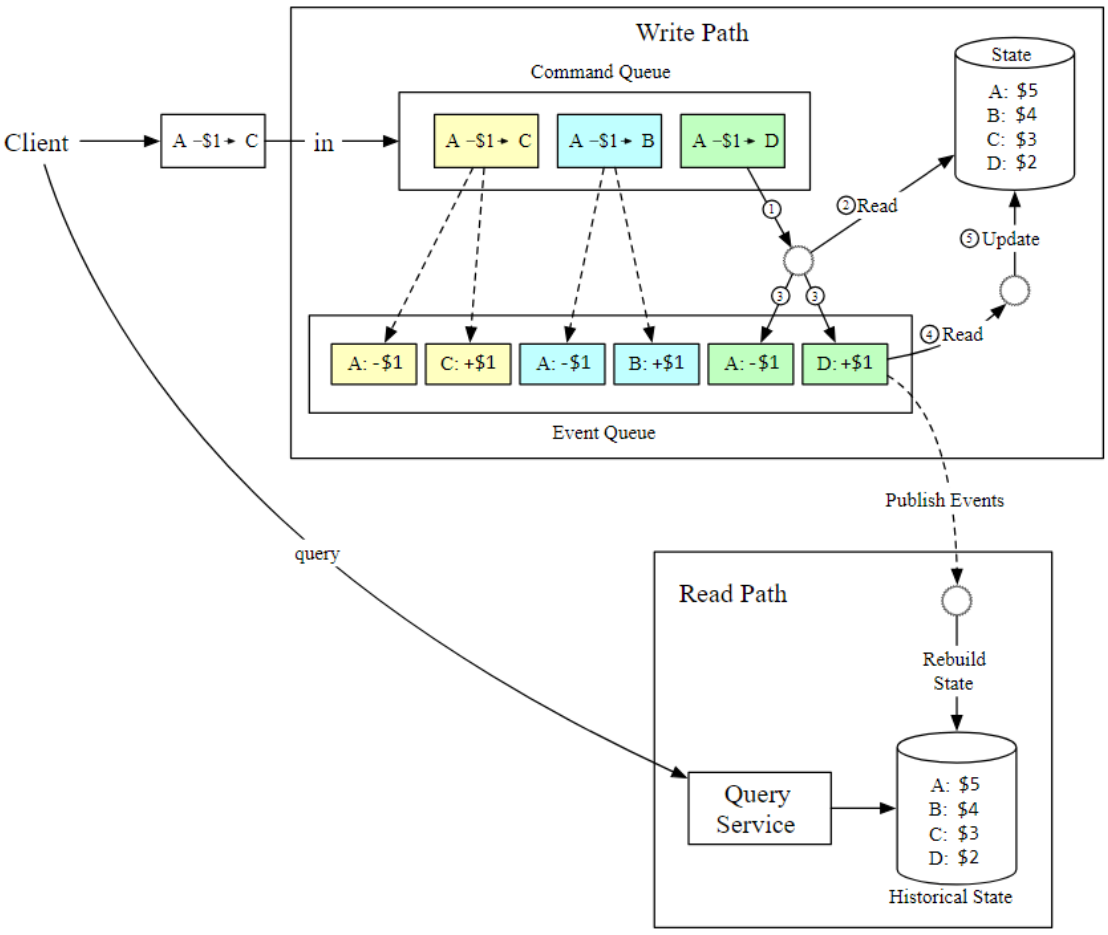


Рисунок 0.11 - CQRS

3.3 Поглиблення у дизайн системи

Файлові команди та список подій

Перша оптимізація полягає у збереженні команд і подій на локальному диску, а не у віддаленому сховищі, як Kafka. Це дозволяє уникнути часу

проходження через мережу. Список подій використовує структуру даних лише для додавання. Додавання — це послідовна операція запису, яка зазвичай дуже швидка. Він добре працює навіть для магнітних жорстких дисків, оскільки операційна система оптимізована для послідовного читання та запису. Відповідно до ACM Queue послідовний доступ до диска може бути швидшим, ніж довільний доступ до пам’яті в деяких випадках.

Друга оптимізація полягає в кешуванні останніх команд і подій у пам’яті. Як ми пояснювали раніше, ми обробляємо команди та події відразу після їх збереження. Ми можемо кешувати їх у пам’яті, щоб заощадити час на завантаження з локального диска.

Ми збираємося дослідити деякі деталі впровадження. Метод під назвою mmap чудово підходить для реалізації оптимізації, згаданої раніше. Mmap може записувати на локальний диск і одночасно кешувати останній вміст у пам’яті. Він відображає файл диска в пам'яті як масив. Операційна система кешує певні розділи файлу в пам’яті, щоб прискорити операції читання та запису. Для операцій додавання файлів майже гарантовано, що всі дані будуть збережені в пам’яті, що дуже швидко.

На малюнку 3.12 показано файлове зберігання команд і подій.

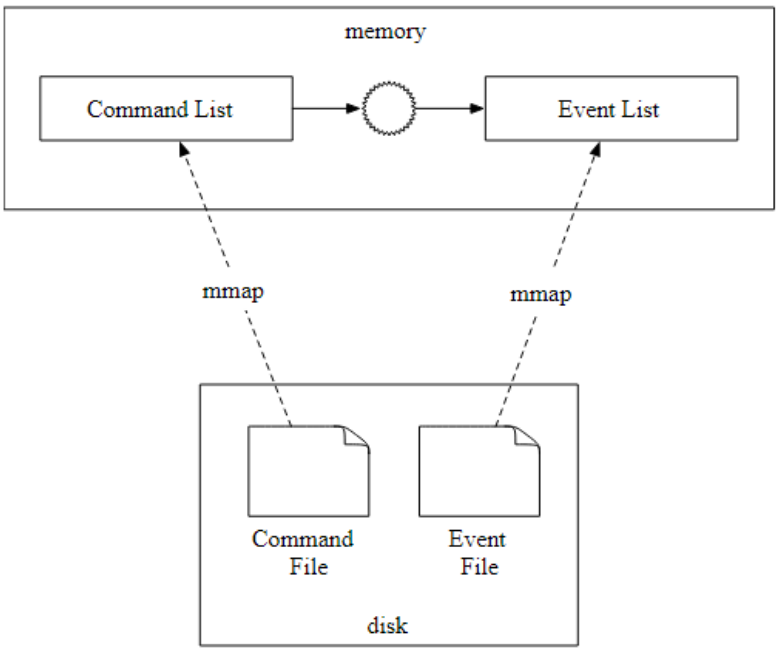


Рисунок 0.12 - зберігання подій у пам’яті

Снепшот

Коли все буде створено на основі файлів, давайте розглянемо, як прискорити процес відтворення. Коли ми вперше запровадили відтворюваність, стейт-менеджер повинен був обробляти події з самого початку, кожен раз. Що ми могли б оптимізувати, так це періодично зупиняти його і зберігати поточний стан у файлі. Це називається знімком.

Знімок — це незмінний погляд на історичний стан. Після збереження моментального знімка менеджер стейту більше не потрібно перезапускати з самого початку. Він може зчитувати дані зі знімка, перевіряти, де він зупинився, і відновлювати обробку звідти.

Для фінансових програм, таких як служба гаманців, фінансова команда часто вимагає, щоб знімок був зроблений о 00:00, щоб вони могли перевірити всі транзакції, які відбулися протягом цього дня. Коли ми вперше представили CQRS джерела подій, рішення полягало в тому, щоб налаштувати менеджер стану лише для читання, який читає з початку, доки не настане вказаний час. З моментальними знімками стейт-менеджеру, доступном лише для читання, потрібно завантажити лише один знімок, який містить дані.

Знімок — це гігантський бінарний файл, і поширеним рішенням є збереження його в системі зберігання об’єктів, такій як HDFS.

3.4 Вибір технологій та підходів з акцентом на безпеку

Оскільки наша головна задача зробити веб-додаток безпечним ми будемо вибирати технології з оглядом на це, не забуваючи про юзер і девелопер-експіріенс.

Зазвичай, цифрові гаманці є великими і складними у розробці веб-додатками, для фронтенд частини(відображення) ми оберемо джава-скрипт фреймворк від компанії Alphabet(Google) - Ангуляр. Він має найбільший вбудований функціонал серед усіх фронтенд технологій: спрощену обробку форм, сервер-сайд рендеринг(для покращення швидкості відклику і загрузки веб-

сторінок), вбудовані інструменти для забезпечення безпеки додатку та його оптимізації, та ще багато іншого.

Давайте виділимо основні фронтенд сценарії, які потрібно покрити:

1. HTTPS і Secure cookies

Спочатку HTTP був запропонований Тімом Бернерсом-Лі, який розробив прикладний протокол для виконання функцій передачі даних високого рівня між веб-серверами та клієнтами. Він використовує добре відомий і зрозумілий протокол HTTP і просто накладає поверх нього рівень шифрування SSL/TLS. Сервери та клієнти все ще спілкуються один з одним за тим самим протоколом HTTP, але через безпечне з'єднання SSL, яке шифрує та розшифровує їхні запити та відповіді. Також усі cookies, які ми будемо використовувати повинні бути http only, завдяки цьому у зловмисника не буде до них доступу з браузера жертви.

2. Санітарна обробка та валідація текстових полів і форм.

Кожне поле у формі є вразливістю, якщо його неправильно обробляти. Обидві ці функції підтримує Ангуляр. Для валідації ми використаємо спеціальні функції валідатори, а для санітарної обробки, яка запобігає XSS та SQL injection атакам використаємо DOMSanitizer клас, який також вбудований у Ангуляр

3. Авторизація, аутентифікація

Автентифікація - це процес перевірки облікових даних користувача або пристрою, які намагаються отримати доступ до обмеженої системи. Тим часом авторизація - це процес перевірки, чи дозволено користувачеві або пристрою виконувати певні завдання в даній системі. Автентифікація виконується лише один раз за всю сессію, коли юзер логіниться, а авторизація відбувається у кожному запиті, який може зчитувати/редагувати/видаляти данні, які повинні бути недоступні для звичайного користувача.

Існує два основних види аутентифікації: JWT і Session. Якщо коротко, то перший більш простий і меньш надійний, другий більш складний і надійний але залишає меньше простору для масштабування. Тому ми використаємо їх обоє, генеруючи два JWT токени: refresh і access, access токен будемо відправляти для перевірки на сервері у авторизаційних заголовках кожного запиту, а refresh токен зберігатимемо

у htmlOnly кукі, що дасть нам змогу мати сессії і не дозволить злочинцям мати доступ до рефреш токену зі сторони клієнта. Додатково ми запровадимо двух-факторну автентифікацію, щоб навіть якщо злочинець отримає доступ до логіну та паролю, то без персонального пристрою користувача він не мав би доступу до гаманця

Для бекенду ж ми запровадимо такі практики:

1. Шифрування і хешування всіх чутливих даних: паролі, дані банківських карток, токени, ключі і т.д.. У випадку отримання доступу до бази даних злочинець матиме на руках тільки нерозбірливий текст замість чутливих даних.

2. Реалізуємо бекенд функціонал ідентифікації, авторизації і автентифікації, про плюси якого ми говорили у фронтенд секції. Він буде включати в себе сервіси для: створення, хешування, внесення, видалення токенів, користувачів та їх ролей на веб-сайті. Також ми додамо функції для перевірки валідності юзера на всі точки доступу до бекенду(endpoints).

3. Для забезпечення надійності і безпеки баз даних ми будемо використовувати зовнішні сервіси для моніторингу, створенню бекапів, шардингу, міграцій і балансування навантаження між копіями баз даних, а саме веб сервіси Amazon: S3, DynamoDB, Cloudfront, Gateway, Cloudwatch і інші.

Висновок: ми розробили повноцінний дизайн веб-системи, яка відповідає таким основним критеріям веб-додатку: масштабованість, надійність, підтримуваність, модульність, безпека та інші. Маючи потрібні навички та користуючись вказівками з цієї секції можна

Висновки

У третьому розділі ми зосереджуємось на розробці надійного веб-додатку, зокрема на роз’ясненні проблеми та визначенні обсягу роботи, розробці поверхневого дизайну системи та його поглибленні. Спочатку ми з'ясували проблему і визначили обсяги робіт. Цей крок включає визначення ключових

системних вимог, виявлення потенційних загроз і вразливостей, а також встановлення критеріїв успіху проекту.

Це дозволяє чітко визначити масштаби розробки та забезпечити цілеспрямоване виконання всіх необхідних етапів роботи. Далі ми розробили поверхневий дизайн системи, включаючи дизайн API, рішення сегментації в пам’яті та застосування методу джерела подій. Під час розробки API ми створили інтерфейси, які забезпечують безпечну та ефективну взаємодію між різними компонентами системи. Рішення для сегментації пам’яті оптимізує керування пам’яттю та забезпечує стабільну роботу додатків навіть за високого навантаження. Використання підходу Event Sourcing зберігає історію змін, внесених у систему, і підвищує надійність і масштабованість системи.

Нарешті, ми глибше заглибилися в дизайн системи, детальніше розглянувши архітектурні рішення, алгоритми та методи захисту, які забезпечують високий рівень безпеки та продуктивності веб-додатку. Цей крок включає аналіз і вибір оптимальних технологій, розробку схеми бази даних та інтеграцію тестування безпеки та інструментів моніторингу.

Отже, у третьому розділі ми застосували комплексний підхід до розробки надійного веб-додатку, який охоплює всі ключові аспекти від визначення проблем і вимог до детального проектування та розгортання технологічних рішень. Це забезпечує безпечну, ефективну та масштабовану систему, готову до використання в реальних середовищах.

ВИСНОВКИ

У даній дипломній роботі було проведено комплексне дослідження сучасних методів захисту веб-застосунків від кіберзагроз. Розглядалися основні типи кібератак, включаючи SQL ін'єкції, Cross-Site Scripting (XSS), Denial of Service (DoS), Man-in-the-Middle (MitM) атаки та Cross-Site Request Forgery (CSRF). Було детально вивчено їх механізми, способи реалізації та потенційні наслідки для веб-застосунків. Також було розглянуто основи криптографічного захисту даних, включаючи симетричне та асиметричне шифрування, хешування та цифрові підписи, а також роль SSL/TLS сертифікатів у забезпеченні безпеки веб-застосунків. Було проаналізовано різні моделі безпеки веб-застосунків, такі як модель загроз та оцінка ризиків, що дозволило зрозуміти, як ідентифікувати та мінімізувати потенційні загрози. Оглянули регулятивні вимоги та стандарти безпеки, такі як GDPR, PCI DSS та ISO/IEC 27001, які встановлюють вимоги до захисту даних та забезпечують правові рамки для їх реалізації.

Окрім теоретичних аспектів, було проведено аналіз існуючих рішень у сфері кібербезпеки, зокрема інструментів, таких як WAF (Web Application Firewalls), RASP (Runtime Application Self-Protection) та IDS/IPS системи (Intrusion Detection/Prevention Systems). Проведено порівняльний аналіз їх ефективності та розглянуто використання штучного інтелекту в системах IDS/IPS. Окремо було проаналізовано випадки найвідоміших кібератак, що дозволило зрозуміти практичні аспекти захисту веб-застосунків.

Також описали процес розробки надійного веб-застосунку з акцентом на безпеку. Включено розробку дизайну системи, вибір технологій та підходів, які забезпечують високий рівень захисту. Було розглянуто використання сучасних методів та інструментів для забезпечення безпеки на кожному етапі розробки, що дозволяє створити захищений та ефективний веб-застосунок.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. A. Hall, C. S. Wright. Data security: a review of major security breaches between 2014 and 2018. 2018.

2. Attacks | OWASP Foundation. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation.

3. Buffard, Sally. The General Data Protection Regulation – a practical guide for trade unionists. Labour Research Department Booklets, 2018.

4. C. Bansal, K. Bhargavan, S. Maffeis. Discovering concrete attacks on website authorization by formal analysis. 2014.

5. Cyber threats 2021: a year in retrospect. PwC.

6. Dan Boneh, Glenn Durfee. Cryptanalysis of Low-Exponent RSA.

7. D. D. Bertoglio, G. Girotto, C. V. Neu. Pentest on an internet mobile app: a case study using tramonto. 2021.

8. D. D. Data. A model to investigate the security challenges and vulnerabilities of cloud computing services in. J. Univ. Shanghai Sci. Technol., 2022.

9. D. Mitropoulos, P. Louridas, M. Polychronakis. Defending against web application attacks: approaches, challenges and implications. 2017.

10. D. S. Berman, J. S. Chavis, A. L. Buczak. A survey of deep learning methods for cyber security. 2019.

11. F. Kong. Research on security technology based on WEB application. 2017.

12. Hardjono. Security In Wireless LANS And MANS. Artech House Publishers, 2005.

13. J. S. Najeem, P. Krishnan. Advanced defence mechanisms for future network security using SDN. 2019.

14. L. Sha, F. Xiao, W. Chen. IIoT-SIDefender : Detecting and defense against the sensitive information leakage in industry IoT. 2017.

15. Marchette D. J., Verma R. M. Cybersecurity analytics. CRC Press LLC, 2022.

16. Nahari H., Krutz R. L. Web commerce security. Wiley & Sons, Incorporated, John, 2012.

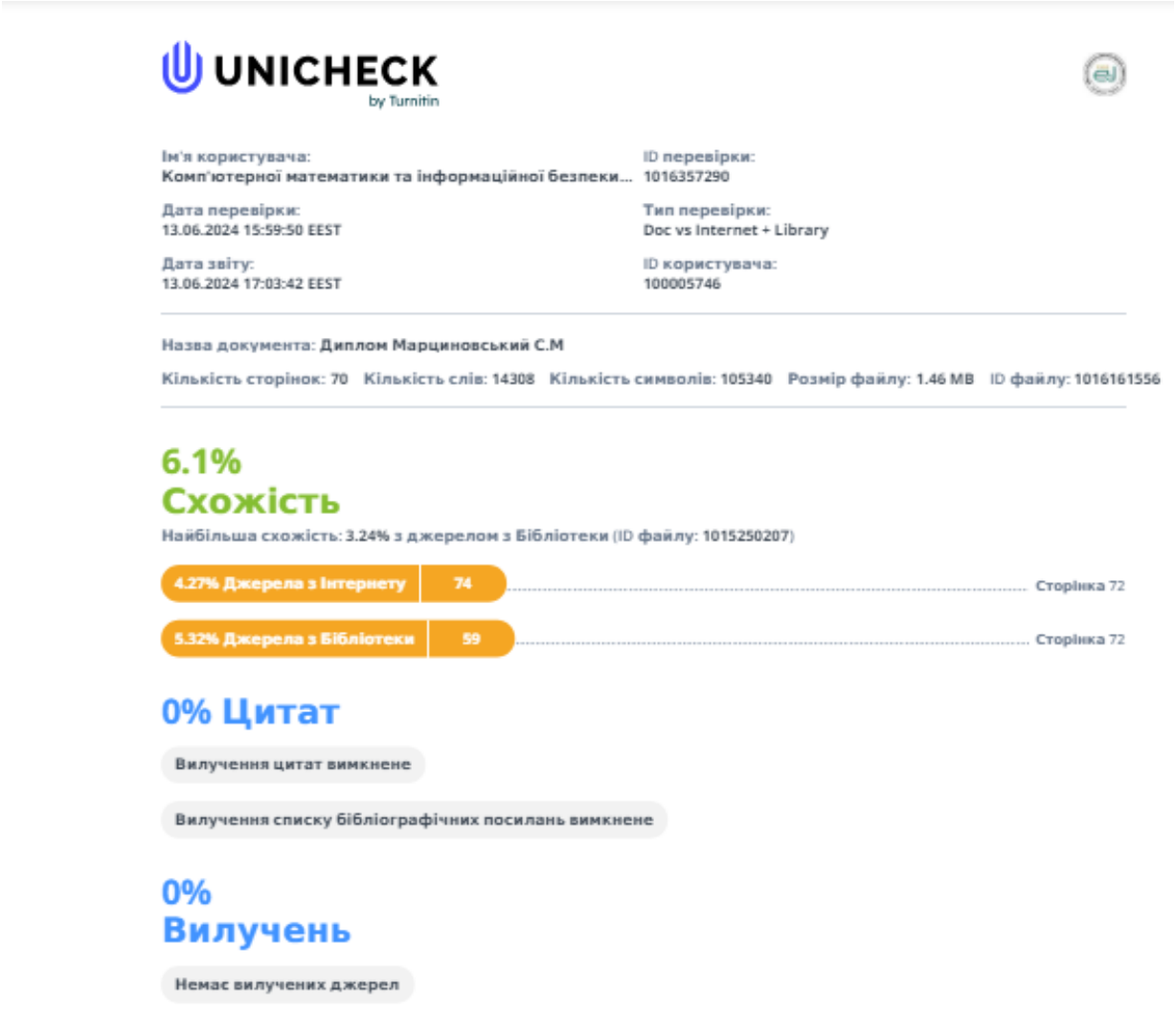
17. S. Chain, S. Mumtaz, M. A. Violas. Man-In-The-Middle attacks in industry. 2019.

18. V. Geetha, P. V. Kallapur. Web security: a survey of latest trends in security attacks. 2011.

19. V. Prajapati, D. Upadhyay. Cyber defence A hybrid approach for information gathering and vulnerability assessment of web application (cyberdrone). Int. J. Comput. Sci. Eng., 2019.

20. XSSed | Cross Site Scripting (XSS) attacks information and archive.
XSSed | Cross Site Scripting (XSS) attacks information and archive.

ЗВІТ ПОДІБНОСТІ



UNICHECK
by Turnitin

Ім'я користувача:
Комп'ютерної математики та інформаційної безпеки...

ID перевірки:
1016357290

Дата перевірки:
13.06.2024 15:59:50 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
13.06.2024 17:03:42 EEST

ID користувача:
100005746

Назва документа: Диплом Марциновський С.М

Кількість сторінок: 70 Кількість слів: 14308 Кількість символів: 105340 Розмір файлу: 1.46 MB ID файлу: 1016161556

6.1%
Схожість

Найбільша схожість: 3.24% з джерелом з Бібліотеки (ID файлу: 1015250207)

4.27% Джерела з Інтернету 74 ..... Сторінка 72

5.32% Джерела з Бібліотеки 59 ..... Сторінка 72

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел