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

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра

студента *Грицюка Сергія Ігоровича*

академічної групи *125м-20-2*

спеціальності *125 Кібербезпека*

спеціалізації

за освітньо-професійною програмою *Кібербезпека*

на тему *Вибір методики тестування захищеності web –додатків від мережевих атак з можливістю обходу міжмережевих екранів що заходяться в середовищі Інтернет*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	к.т.н., доц. Флоров С.В			
розділів:				
спеціальний	к.т.н., доц. Флоров С.В			
економічний	к.е.н., доц. Пілова Д.П.			

Рецензент				

Нормоконтролер	ст. викл. Тимофєє Д.С.			

Дніпро
2022

ЗАТВЕРДЖЕНО:

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

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу ступеня магістра

студенту *Грицюка Сергія Ігоровича* **академічної групи** *125м-20-2*
(прізвище та ініціали) (шифр)

спеціальності *125 Кібербезпека*

спеціалізації

за освітньо-професійною програмою *Кібербезпека*

на тему ***Вибір методики тестування захищеності web –додатків від мережевих атак з можливістю обходу міжмережевих екранів що заходяться в середовищі Інтернет***

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від _____ № _____

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень *процес тестування захищеності WEB-додатків за допомогою методів «чорного» ящика*

Предмет досліджень *методика тестування захищеності WEB-додатків на вразливість HTTP Parameter Pollution.*

Мета *розробка методики тестування WEB-додатків на вразливість HTTP та обґрунтування ефективності використання запропонованої методики.*

Вихідні дані проведення роботи *результати та матеріали, переддипломної практики та курсовому проекту з комплексних систем захисту інформації.*

3 ОЧІКУВАНІ РЕЗУЛЬТАТИ

Наукова новизна *полягає у вирішенні проблеми тестування захищеності WEB-додатків від атак зі сторони клієнта на основі використання методики*

що включає в себе використання ПЗ та виконання тестування на проникнення.

Практична цінність *полягає у розроблені методики тестування WEB-додатків на вразливість HTTP та аналізу атак на стороні клієнта із використанням вразливості HTTP.*

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати роботи мають відповідати вимогам чинного законодавства України та бути поданим у вигляді, що дозволяє безпосереднє використання при прийнятті рішення про застосування хмарних технологій в учбових закладах**.**

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт** (початок-кінець)
Огляд джерел за темою та напрям досліджень	03.10.21-16.10.21
Методи досліджень	17.10.21-31.10.21
Результати досліджень	01.11.21-22.12.21
Виконання економічного розділу	23.12.21-04.01.21
Оформлення пояснювальної записки	05.01.22-10.01.22

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект *від реалізації результатів роботи очікується позитивним завдяки визначенню найбільш актуальної загрози інформації з завдяки визначенню найбільш актуальної загрози інформації.*

Соціальний ефект *від реалізації результатів роботи очікується позитивним завдяки розробці методики тестування захищеності WEB-додатків перевагою якого є можливість обходу міжмережевого екрану*

7 ДОДАТКОВІ ВИМОГИ

Відповідність оформлення «ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення» та «Методичні вказівки. Загальні вимоги до оформлення магістерських дипломних робіт і дипломних проектів спеціалістів для студентів галузей знань 1701 «Інформаційна безпека» та 0509 «Радіотехніка, радіоелектронні апарати та зв'язок»

Завдання видано _____ Флоров С.В.
(підпис керівника) (прізвище, ініціали)

Дата видачі: 03.10.21р.

Дата подання до екзаменаційної комісії: 14.01.21р.

Прийнято до виконання _____ Г'рицюк С.І.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка: с., 27 рис., 12 табл., 4 додатка, 30 джерел.

Мета магістерської дипломної роботи: обґрунтувати вибір методики тестування захищеності web – додатків від мережевих атак з можливістю обходу Web Application Firewall та практично довести ефективність запропонованої методики.

У розділі «Стан питання. Постановка задачі» було розглянуто вразливості WEB-додатків та особливості функціонування протоколу HTTP 2.0.

У спеціальній частині проаналізовано вразливість HTTP Parameter Pollution на стороні клієнта, основною перевагою якого є можливість обходу WAF (Web Application Firewall), обрано та забезпечено профіль захищеності, розроблена методика тестування захищеності WEB-додатків, практично реалізовано тестування захищеності тестового додатку.

У економічній частині виконано розрахунок вартості заходів щодо інтеграції програмних заходів тестування WEB-додатків, що використовуються у методиці, а також розрахунок збитку від атак зі сторони клієнта на обчислювальну мережу WEB-додатка. Надано оцінку економічної доцільності впровадження методики тестування захищеності.

Наукова новизна роботи полягає в розробці методики тестування захищеності WEB-додатків на вразливість HTTP Parameter Pollution, основною перевагою якого є можливість обходу WAF (Web Application Firewall.

WEB-ДОДАТОК, ТЕСТУВАННЯ ЗАХИЩЕНОСТІ, ВРАЗЛИВІСТЬ HTT, WEB APPLICATION FIREWALL.

РЕФЕРАТ

Пояснительная записка: с., 27 рис, 12 табл, 4 приложения, 30 источников.

Цель магистерской дипломной работы: обосновать выбор методики тестирования защищенности web – приложений от сетевых атак с возможностью обхода Web Application Firewall и практически доказать эффективность предложенной методики.

В разделе «Состояние вопроса. Постановка задачи» были рассмотрены уязвимости WEB-приложений и особенности функционирования протокола HTTP 1.1.

В специальной части проанализирована уязвимость HTTP Parameter Pollution на стороне клиента, основным преимуществом которого является возможность обхода WAF (Web Application Firewall), выбран и обеспечен профиль защищенности, разработана методика тестирования защищенности WEB-приложений, реализовано тестирование защищенности тестового приложения.

В экономической части выполнен расчет стоимости мероприятий по интеграции программных средств тестирования веб-приложений, которые используются в методике, а также расчет ущерба от атак со стороны клиента на вычислительную сеть WEB-приложения. Дана оценка экономической целесообразности внедрения методики тестирования защищенности.

Научная новизна работы заключается в разработке методики тестирования защищенности WEB-приложений на уязвимость HTTP Parameter Pollution, основным преимуществом которого является возможность обхода WAF (Web Application Firewall.

WEB-ПРИЛОЖЕНИЕ, ТЕСТИРОВАНИЕ ЗАЩИЩЕННОСТИ, УЯЗВИМОСТЬ HTTP, WEB APPLICATION FIREWALL.

ABSTRACT

The explanatory note: pages, 27 pictures, 12 tables, 4 additions, 30 sources.

The purpose of master's thesis: to justify the choice of methods of testing the security of web applications from network-based attacks on Web Application Firewall and almost to prove the effectiveness of the proposed method..

In the "Status of the issue. Formulation of the problem" were considered Web applications vulnerabilities and features of functioning HTTP 1.1 protocol.

In a special part was analyzed HTTP Parameter Pollution vulnerability on the client side, the main advantage of which is the ability to bypass WAF (Web Application Firewall), selected and provided security profile, developed the methodology for security testing of Web applications, implemented security testing of test application.

On the economic side, we calculated the cost for the integration of software for testing Web applications, that is used in the methodology, as well as the calculation of damage from attacks from the client to the computer network of Web applications. Assessed the economic feasibility of implementing the security testing methodology.

Scientific novelty of the work is to develop the methodology for security testing of Web applications from HTTP Parameter Pollution vulnerability, the main advantage of which is the ability to bypass WAF (Web Application Firewall).

WEB-APPLICATION, SECURITY TESTING, HTTP, WEB APPLICATION FIREWALL .

СПИСОК УМОВНИХ СКОРОЧЕНЬ

АС – автоматизована система;

БД – база данних;

КЗЗ – комплекс засобів захисту;

НД – нормативный документ;

НДР – науково-дослідна робота;

НСД – несанкціонований доступ;

ОС – операційна система;

ПЗ – програмне забезпечення;

СЗІ – система захисту інформації;

СКБД – система керування базами даних;

ТЗІ – технічний захист інформації;

DoS – відмова в обслуговуванні (denial of service);

HTTP – гіпертекстовий протокол передачі (hypertext transfer protocol);

IP – інтернет протокол (internet protocol);

ID – ідентифікаційний документ (identity document);

LDAP – полегшений протокол доступ до директорій (lightweight directory access protocol);

SQL – мова структурних запитів (structured query language);

SSI – серверні додатки (server side includes);

TCP – протокол управління передачею (transmission control protocol);

URL – уніфікована адреса ресурсу (uniform resource locator);

WEB – всесвітня мережа (world wide web);

WAF – міжмережевий екран (Web Application Firewall);

XSS – міжсайтовий скриптинг (cross site scripting).

ВСТУП

Вже не перший рік безпека WEB-додатків є одним з ключових елементів захисту інформаційних систем. Актуальність захисту WEB-додатків зростає – цьому свідчить зростаюча тенденція до перенесення стандартних клієнт – серверних додатків у середовище Інтернет, розвиток технології WEB 2.0, а також розвиток різних ланок Інтернет бізнесу.

WEB-додатки є найбільш розповсюдженими механізмом при побудові бізнесу різних сфер діяльності. Саме WEB-додатки дали змогу багатьом компаніям не тільки закріпитися на ринках, але й привабити нових клієнтів. Але окрім потенційних користувачів-клієнтів, WEB-додатки приваблюють різноманітних зловмисників, які мають свою метою отримання конфіденційної інформації.

Світова статистика з інформаційної безпеки свідчить про те, що майже 75% атак на WEB-сайти компаній припадає на WEB-додатки, які вони використовують. Більше половини з цих атак експлуатують вразливості у додатках.

Поширені вразливості WEB-додатків були розглянуті на WEB Application Security Consortium, де була прийнята загальна класифікація загроз, яка складалася з шести класів атак, в кожному з яких були розглянуті найбільш ймовірні типи:

- клас атак, що використовує механізми автентифікації;
- клас атак, що використовує механізми авторизації;
- клас атак, що використовує механізми атаки на клієнтів;
- клас атак, що використовує механізми виконання коду на сервері;
- клас атак, направлених на розголошення інформації;
- клас атак, що використовує механізми логіки роботи WEB-додатка.

Атаки, що базуються на вразливості HTTP Parameter Pollution, основною перевагою якого є можливість обходу WAF (Web Application Firewall), хоча й тільки недавно були зафіксовані, але постійно вдосконалюються. Незважаючи на, перший погляд, просту реалізацію, експлуатація вразливості, що базується на такому розповсюдженому протоколу, як HTTP, може бути основою для використання інших видів не менш загрозливих атак: міжсайтового скриптингу та ін..

Останні дослідження у сфері інформаційної безпеки свідчать про те, що близько 40% WEB-додатків є потенційно вразливими для HTTP Parameter Pollution. Більша половина з вразливих додатків відноситься до фінансових та державних установ. Тому дуже важливим є не тільки локалізація даної вразливості для вже існуючих додатків, але й знаходження та попередження них для нових додатків – на етапі розробки та введення в експлуатацію.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

1.1. Архітектура WEB-додатків

WEB-додаток - це клієнт – серверний додаток, в якому сервером виступає WEB-сервер, а клієнтом виступає WEB-браузер, що знаходиться на стороні користувача. Логіка роботи WEB-додатку є розподіленою між сервером та клієнтом. Зберігання даних здійснюється на сервері, обмін інформацією відбуваються по мережі. До головних переваг такої архітектури можна віднести міжплатформенність WEB-додатків – клієнти не залежать від конкретної операційної системи.

Клієнтська частина WEB-додатка реалізує інтерфейс користувача, формує запити до сервера і обробляє відповіді від нього.

Серверна частина отримує запит від клієнта, виконує обчислення, після цього формує WEB-сторінку і відправляє її клієнту по мережі з використанням протоколу HTTP.

Архітектура WEB-додатка містить:

- клієнтську частину;
- серверну частину;
- канал передачі даних.

Клієнтською частиною є браузер. За допомогою браузеру користувач має можливість звертатися до форм WEB-додатків, а також відправляти та приймати дані.

Канал передачі даних представляє транспортний засіб по Internet за допомогою стеку протоколів TCP/IP [1].

Серверна частина WEB-додатка може бути представлена:

- WEB-серверами;

- серверами додатків;

- серверами баз даних;
- файл-серверами;
- проксі-серверами;
- firewall'ами;
- поштовими серверами.

1.2. Логіка побудови WEB-додатків

На даний момент існує та успішно застосовується різні види технологій побудови WEB-додатків. Усі такі додатки мають загальну ціль – реалізацію бізнес – логіки на стороні серверу та генерацію коду для клієнту. Також для усіх додатків є однаковою взаємодія серверу та клієнту, а також спільний протокол взаємодії – HTTP.

На рисунку 1.1 зображена схема роботи WEB-додатка.

1. Користувач ініціює запит до WEB- сервери використовуючи для цього свій WEB-браузер.
2. WEB-сервер перенаправляє отриманий запит до доступного серверу WEB-додатків.
3. Сервер WEB-додатку виконує задачу, отриману у запиті.
4. Сервер WEB-додатку отримує доступ до серверу баз даних.
5. Сервер WEB- додатку формує відповідь та повертає її до WEB-серверу.
6. WEB-сервер посилає відповідь користувачеві, яка містить дані про успішність операції та дані, що запитувалися.
7. Інформація з'являється на моніторі користувача.

До основних недоліків подібної схеми роботи можна віднести роботу тільки в режимі запит – відповідь, тобто не має даних про попередні кроки користувача або постійної інформації.

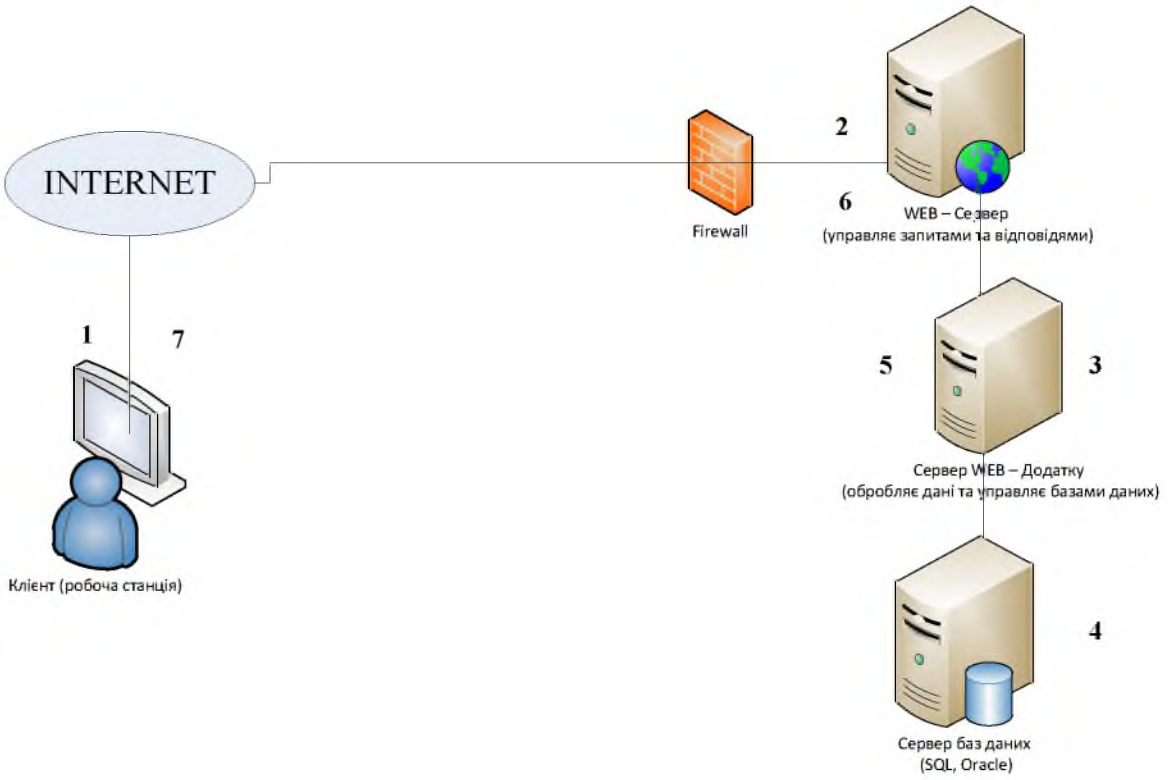


Рисунок 1.1 - Архітектура та логіка роботи WEB – додатка

WEB-додаток розробляється на одній із серверних мов програмування. Скомпільовані варіанти розміщуються на окремому сервері додатків, доступ до якого має WEB-сервер, який обробляє клієнтські запити. Сервер додатків в свою чергу має доступ до серверів баз даних.

Логіка роботи WEB-додатку ґрунтується на формуванні запитів на клієнтській стороні, передачі їх по каналу даних (у формі HTTP-запитів) та подальшій обробці на WEB-сервері, сервері додатків, який в свою чергу має можливість взаємодіяти с серверами баз даних для отримання потрібної інформації.

Трирівнева модель обробки запитів WEB-додатком є найпростішою моделлю для побудови всіх додатків.

Багаторівнева система обробки запитів дозволяє працювати одночасно з різними частинами серверної частини і отримувати доступ до окремих сегментів управління WEB-сервером, сервером додатків та сервером БД.

Особливістю такої моделі, з точки зору інформаційної безпеки, є те, що некоректний запит з боку клієнта може вплинути на працездатність всієї

серверної частини програми або у випадку неправильної авторизації, надати певні права доступу на роботу з серверами БД.

Вразливості у WEB-додатках в більшості випадків виникають через недостатню перевірку введених даних. Отже, існує проблема, пов'язана з контролем і коректністю введення даних на стороні клієнта, запити з якої створюють ряд загроз для WEB-додатків [2].

1.3. WEB – сервери

WEB-сервер - це мережевий додаток, що обслуговує HTTP-запити від клієнтів, зазвичай веб-браузерів. Веб-сервер приймає запити і повертає відповіді, зазвичай разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними. WEB-сервери - основа Всесвітньої павутини.

Простий обмін між клієнтом та веб – сервером виглядає таким чином [3]:

1 Браузер користувача аналізує URL – адресу та виділяє окремі частини – шлях, версію протоколу та інше.
2 Сервер доменних імен (DNS) перетворює доменне ім’я WEB-сайту, на який зайшов користувач, на IP – адресу.
3 Браузер користувача визначає, який протокол потрібно використовувати – HTTP або FTP та якої версії.
4 Браузер посилає запит до WEB-серверу,в якому визначається які дані потрібно передати користувачу.
5 WEB-сервер відповідає на запити браузера. Він підтверджує, що дана адреса існує, знаходить потрібні файли, запускає відповідні сценарії та повертає результати назад до браузеру. У разі, коли файл не може бути знайдений, або команду запит неможливо виконати, WEB-сервер відправляє повідомлення про помилку користувачу.
6 Браузер переводить дані, що були отримані від WEB-серверу у HTML та відображає користувачеві.

Створенням програмного забезпечення WEB-серверів займаються багато розробників, але найбільшу популярність в WWW отримали такі програмні продукти, як Apache (Apache Software Foundation), IIS (Microsoft), QZHTTP (він же qq.com), Google Web Server (GWS, Google Inc .) і nginx (Рисунок 1.2).

Рисунок 1.2 – Ринок WEB- серверів, Netcraft

Apache - безкоштовний WEB - сервер з відкритим вихідним кодом, розповсюджується під сумісною з GPL ліцензією. Apache вже багато років є лідером по поширеності у Всесвітній павутині в силу своєї надійності, гнучкості, масштабованості і безпеки.

IIS (Internet Information Services) - пропріетарний набір серверів для декількох служб Інтернету, розроблений Microsoft і поширюваний з ОС сімейства Windows NT. Основним компонентом IIS є WEB-сервер, також підтримуються протоколи FTP, POP3, SMTP, NNTP.

QZHTTP - модифікований Apache, який використовується на китайському порталі qq.com. На ньому розміщені сервіси онлайн-щоденників і миттєвого обміну повідомленнями.

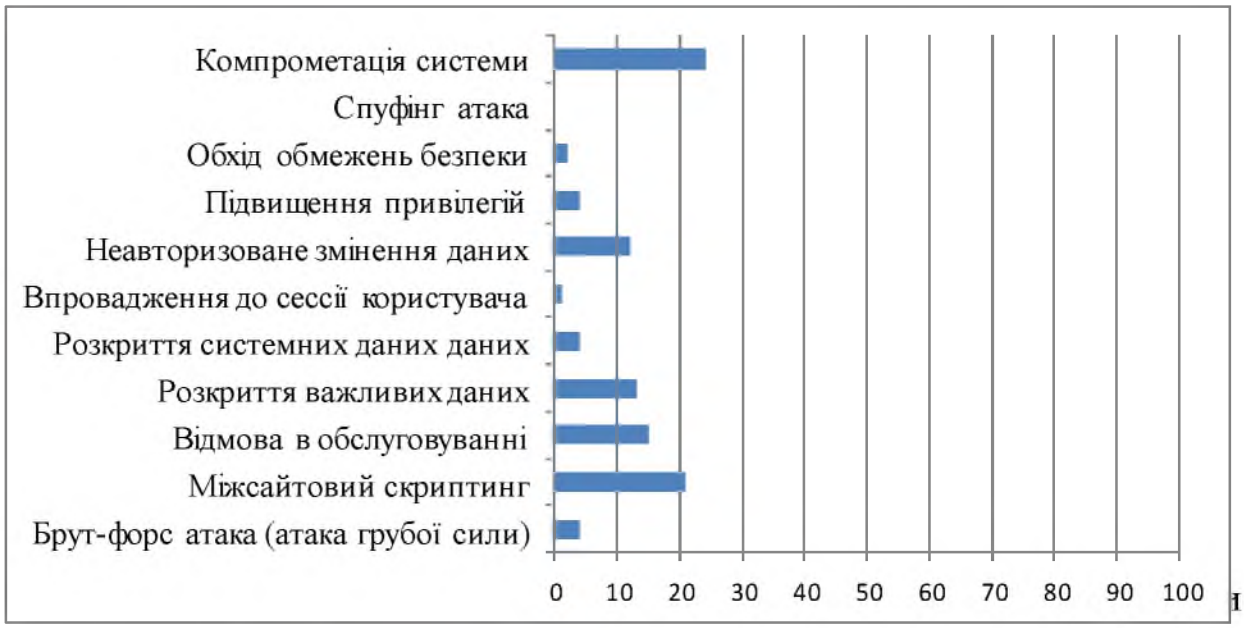
Google Web Server (GWS) - розробка компанії Google на основі WEB-сервера Apache. GWS оптимізований для виконання додатків сервісу Google Applications.

nginx – це HTTP-сервер, сполучений з поштовим проксі-сервером. Розроблено І.Сисоевим для компанії Рамблер. Восени 2004 року вийшов перший публічно доступний реліз, зараз nginx використовується низкою великих сайтів.

lighttpd - WEB-сервер, що розробляється з розрахунком на швидкість і захищеність при використанні на сильно навантажених сайтах, а також відповідність стандартам. lighttpd - безкоштовне програмне забезпечення, яке розповсюджується за ліцензією BSD [4].

1.4. Класифікація атак на WEB-додатки

На рисунку 1.3 зображена статистика використання вразливостей WEB-додатків за 2015 рік, зібрана компанією Positive Technologies [5].



висновок, що лідируючи позиції серед вразливостей станом на 2015 рік займають компрометація системи, міжсайтовий скриптинг та відмова у обслуговуванні.

Класифікація атак на WEB-додатки має ієрархічну структуру та розділяється на шість основних класів. Класи атак наведені на рисунку 1.4:

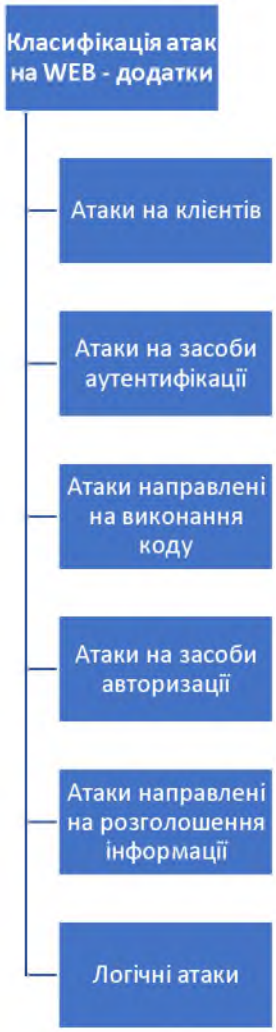


Рисунок 1.4 – Класифікація атак на WEB-додатки

1.4.1 Атаки на клієнтів

Експлуатуючи довіру, що виникає між користувачем сайту та сервером під час експлуатації додатку, зловмисник має можливість використовувати різні методи для проведення атак на клієнтів сервера.

Міжсайтове виконання сценаріїв

Наявність даної вразливості дозволяє зловмиснику передавати серверу виконуваний код, який у подальшому буде пере направлений до WEB-браузеру користувача. Зловмисний код може бути написаний за допомогою HTML/JavaScript, VBScript, ActiveX, Java, Flash та ін. Зловмисні дії коду полягають у можливості читання, модифікації або передачі даних, доступних для WEB-браузеру.

Використовуючи вразливість цього типу зловмисник може скомпрометувати аккаунт користувача, пере направити WEB-браузер на інший сервер або виконати підміну вмісту WEB-сайту.

При міжсайтовому виконанні сценаріїв передача коду здійснюється через URL, у заголовках HTTP-запиту, значеннях полів форм. Розрізняють два типи атак: постійні (бережені) та непостійні (відбиті). Головна відмінність між цими атаками полягає у тому, що у відбитих атаках передача коду серверу здійснюється у рамках одного HTTP-запиту, а в збереженому – в різних.

Збережені вразливості виникають у випадку передачі коду серверу та зберіганні на ньому деякий проміжок часу. Найчастіше дані вразливості використовуються на форумах, чатах та поштах.

На рисунку 1.5 зображено приклад реалізація міжсайтового сккриптингу:

```
aaa@aa.com"><script>alert(document.cookie)</script>
```

Рисунок 1.5 – Приклад міжсайтового скрптингу

Приклад після URL-кодування (рисунок 1.6):

```
aaa@aa.com%22%3E%3Cscript%3Ealert(document.cookie)%3C%2Fscript%3E
```

Рисунок 1.6 - Приклад міжсайтового скрптингу після URL-кодування

Відбиті вразливості виникають у випадку, коли користувач самостійно переходить по посилання, яке було сформовано зловмисником. В процесі завантаження WEB-сайту зловмисний код буде передано WEB-браузеру користувача і виконано [6].

Підміна HTTP-запиту

При використанні даної уразливості зловмисник посилає серверу спеціальним чином сформований запит, відповідь на який інтерпретується

метою атаки як два різні відповіді. Друга відповідь повністю контролюється зловмисником, щодає йому можливість підробити відповідь сервера.

В результаті реалізації атаки зловмисник може виконати наступні дії:

1 Міжсайтове виконання сценаріїв;

2 Модифікація даних кеша сервера-посередника.

Сервери – посередники, що використовуються для доступу на даний WEB-сайт, можуть зберігати підроблену зловмисником відповідь на жорсткому диску. При цьому на всі наступні запити користувачів сервер повертає модифіковані зловмисником дані – а в результаті – виникає підміна сторінок сервера на стороні клієнта.

3 Міжкористувальницька атака.

Суть даної атаки полягає у тому, що сервер – посередник розділяє одне TCP-з'єднання з сервером між декількома користувачами. В результаті другий користувач у відповідь на запит може отримати сторінку, сформовану зловмисником.

4 Перехоплення сторінок із даними, що призначені для користувача.

В цьому випадку зловмисник отримує відповідь сервера замість самого користувача. Таким чином, він може дістати доступ до конфіденційної інформації.

5 Модифікація HTTP – запитів до WEB – сервера.

У цьому випадку зловмисник може модифікувати HTTP – запит або сторінку, з якою взаємодіє сам користувач з метою отримання доступу до конфіденційних даних або виконання дій, що становлять загрозу. Прикладом атак такого типу може бути атака HTTP Parameter Pollution, основною перевагою якого є можливість обходу WAF (Web Application Firewall).

1.4.2 Атаки на засоби аутентифікації

Атак даного класу направлені на експлуатацію вразливостей в механізмах аутентифікації WEB-серверів.

Підбір

Підбір – автоматизований процес спроб на похибок, основною метою якого є вгадування пароля та імені користувача, нормера кредитної картки, та ін..

Основною проблемою для забезпечення механізму аутентифікації є той факт, що користувачі у більшості випадків обирають легковгадувані паролі.

Приклади простих паролів:

password1, *deer2000*, *john1234*,*qwerty*, *12345*, *asdfgh*, *fred*, *stopstop*, *treetree*, *passpass*.

В таких випадких існує потенційна загроза підбору пароля зі сторони зловмисника: використовуючи електронний словник, він може підібрати дані користувача, необхідні для аутентифікації.

Найчастіше підбір використовуєься для отримання ключів шифрування. У тому випадку, коли сервер використовує ключові комбінації недостатньо великої довжини, зловмиснику стає можливим отримати потрібний ключ, просто перебравши усі можливі комбінації.

Існує два види підбору – прямий та зворотній.

Прямий підбір – один варіант імені користувача для різних варіантів паролів.

Зворотній підбір – один варіант пароля для різних імен користувачів

До недоліків даних атак можна віднести те, що в залежності від криптостійкості пароля час на його підбір зростає до декількох днів або років. Тому підбір використовується зазвичай у випадку коли блокування у разі неправильного вводу даних на WEB-сайті відсутнє [7].

Недостатня аутентифікація

Існують випадки, коли WEB-сервер дозволяє зловмиснику мати доступ

до важлвої інформації без належної аутентифікації.

Щоб не використовувати аутентифікацію, деякі ресурси по дефолту використовують певну адресу, яка не вказана на основних сторінках сервера або інших загальнодоступних ресурсах. Необхідний URL може бути

знайдений шляхом перебору типових файлів і директорій (таких, як /admin/) з використанням повідомлень про помилки журналів перехресних посилань або шляхом простого читання документації. Подібні ресурси мають бути захищені адекватно важливості їх вмісту і функціональних можливостей [7].

Небезпечне відновлення паролів

Реалізація даної вразливості довзоляє зловмиснику несанкціоновано отримувати, відновлювати або модифікувати паролі інших користувачів.

Прикладом реалізації функції відновлення паролю є використання «секретного питання», відповідь на який вказується в процесі реєстрації. Питання або вибирається із списку, або вводиться самим користувачем. Ще один механізм дозволяє користувачеві вказати «підказку», яка допоможе йому згадати пароль. Інші способи вимагають від користувача вказати частину персональних даних - таких, як номер паспорта, домашня адреса, поштовий індекс і так далі, - які потім використовуватимуться для встановлення особи. Після того як користувач доведе свою ідентичність, система відобразить новий пароль або перешле його поштою.

Вразливості, що ґрунтуються на недостатній перевірці при відновленні пароля, виникають у тому випадку, коли зловмисник отримує дані, що використовуються механізмом відновлення.

Це можливо, коли інформація для відновлення пароля є легковгадуваною або сам процес відновлення має недоліки – і в результаті його можливо обійти.

1.4.3 Атаки направлені на виконання коду

Логіка роботи WEB-додатку у більшості випадків ґрунтується на даних, що були передані зі сторони клієнта. Крім даних необхідних для авторизації, дані передані користувачем можуть використовуватися для формування запитів та генерації динамічного вмісту WEB-сторінок.

У випадку, коли на етапі розробки WEB-додатку вимоги безпеки не враховуються, зловмисник має можливість модифікувати виконувані команди [7].

Переповнення буфера

Вразливість переповнення буфера – найпоширеніша на даний момент в обасліт безпеки ПЗ. Переповнення виникає у випадку, коли об'єм даних перевищує розмір виділеного під них буфера. Коли буфер переповнюється, дані переписують інші області пам'яті, що призводить до виникнення помилки. Переповнення буфера може викликати відмови в обслуговуванні, призводячи до ушкодження пам'яті і викликаючи помилки в програмах.

Переповнення буфера може викликати відмови в обслуговуванні, змінити шлях виконання програми і виконати в її контексті різні дії, перезаписувати службові області пам'яті, також, при переповненні можуть бути переписані значення змінних у програмі [7].

Атака на функції форматування рядків

При використанні цих атак шлях виконання програми модифікується методом перезапису областей пам'яті за допомогою функцій форматування символьних змінних. Уразливість виникає, коли призначені для користувача дані застосовуються як аргументи функцій форматування рядків - таких, як fprintf, printf, sprintf, setproctitle, syslog і так далі.

Впровадження операторів LDAP

Атаки цього типу спрямовані на WEB-сервери, які створюють запити до служби LDAP на основі даних, що вводяться користувачем. Протокол LDAP працює поверх транспортних протоколів Internet (TCP/UDP).

WEB-сервер використовує дані, надані користувачем, для генерації динамічних сторінок шляхом створення запитів по протоколу LDAP. У тому випадку, коли інформація, отримана від клієнта не перевіряється належним чином, то зловмисник може отримати можливість модифікувати LDAP-запит.

Приклад:

У WEB-додатку існує фільтр, що приймає пару логін/пароль по протоколу LDAP [8].

earchlogin="(&(uid="+user+")(userPassword={MD5}"+base64(pack("H ",md5(pass)))+"))";*

Зловмисник використовує такі параметри запиту:

user=) (uid=*)) (| (uid=* pass=password)*

В результаті фільтр буде повертати:

searchlogin="(&(uid=)(uid=*))(|(uid=*)(userPassword={MD5}X03MO1 qnZdYdgyfeuILPmQ==))";*

Виконання команд ОС

Атаки цього класу спрямовані на виконання команд операційної системи на WEB-сервері шляхом маніпуляції вхідними даними. Якщо інформація, отримана від клієнта, належним чином не верифікуються, атакуючий отримує можливість виконати команди ОС. Вони будуть виконуватися з тим же рівнем привілеїв, з яким працює компонент додатку, що виконує запит (сервер СУБД, WEB-сервер іт.д).

Більшість мов сценаріїв дозволяють запускати команди ОС під час виконання, використовуючи варіанти функції exec. Якщо дані, отримані від користувача передаються цій функції без перевірки, зловмисник може виконати команди ОС на відстані.

Впровадження операторів SQL

Атаки даного типу використовують WEB-сервери, що створюють SQL-запити до серверів СКБД, що формуються на основі даних користувача.

Мова запитів SQL є спеціалізованою мовою програмування, що дозволяє створювати запити до серверів СКБД. Більшість серверів підтримують цю мову у варіантах, стандартизованих ISO і ANSI. У більшості сучасних СКБД присутні розширення діалекту SQL специфічні для цієї реалізації (T-SQL в Microsoft SQL Server, -PL SQL в Oracle і т. д. Якщо інформація, отримана від клієнта, належним чином не верифікуються, атакуючий отримує можливість модифікувати запит до SQL-серверу, що відправляється додатком. Запит буде виконуватися з тим же рівнем привілеїв, з яким працює компонент додатку, що виконує запит (сервер СКБД, WEB-сервер і т.д). У результаті зловмисник може отримати повний контроль над сервером СУБД і навіть його операційною системою.

Розрізняють два основних методи експлуатації операторів SQL: звичайна атака та атака всліпу.

Звичайна атака – виконується підбір параметрів запиту на основі інформації про помилки, що були згенеровані WEB-додатком.

Приклад:

Правильний запит до СКБД [9]:

*SELECT * FROM Users WHERE Username='$username' AND Password='$password'*

Модифікований запит до СКБД:

*SELECT * FROM Users WHERE Username='1' OR '1' = '1' AND Password='1' OR '1' = '1'*

Модифікований запит до WEB-додатку:

http://www.example.com/index.php?username=1'%20or%20'1'%20=%20'1 &password=1'%20or%20'1'%20=%20'1

Атака всліпу – додавання до запитів виразів, що завжди повертають істинне або помилкове значення.

Приклад:

Модифікований запит до WEB-додатку:

http://www.example.com/index.php?id=1'

Впровадження серверних розширень

Атаки даного класу дозволяють зловмиснику передати виконуваний код, який надалі буде виконаний на WEB-сервері. Уразливості, що приводять до можливості здійснення даних атак, зазвичай полягають у відсутності перевірки даних, наданих користувачем, перед збереженням їх у скриптової сервером файлі.

Якщо атакуючий передає серверу оператори SSI, він може отримати можливість виконання команд операційної системи або включити до неї заборонене вміст при наступному відображенні.

Впровадження операторів XPath

Ці атаки спрямовані на WEB-сервера, які створюють запити на мові XPath на основі даних, що вводяться користувачем. Мова XPath 1.0 розроблена для надання можливості звернення до частин документу на мові XML. Він може бути використаний безпосередньо або як складова частина XSLT-перетворення XML-документів, або як виконання запитів XQuery. Синтаксис XPath близький до мови SQL-запитів.

Приклад:

Правильний запит [10]:

string(//user[username/text()='gandalf' andpassword/text()='!c3']/account/text())

Модифікований запит:

string(//user[username/text()='' or '1' = '1' and password/text()='' or '1' = '1']/account/text())

1.4.4 Атаки на засоби авторизації

Процес авторизації полягає у визначенні чи має користувач дозвіл на здійснення тієї чи іншої дії. Більшість WEB-ресурсів мають декілька типів користувачів, кожен із яких має свій спектр дозволених дій. Доступ до дій та даних, що не входять до списку дозволених повинен бути обмежений. Головним завданням зловмисника у цьому випадку є підвищення власних привілеїв для отримання доступу до захищених даних.

Недостатня авторизація

Недостатня авторизація полягає у тому, що WEB-сервер дозволяє зловмиснику мати доступ до інформації або функції, які повинні бути обмежені для цього користувача. Процедура авторизації направлена на розмежування доступу до WEB-ресурсу. Правила доступу повинні бути чітко вказані – повинна виконуватися політика безпеки. Доступ до важливих даних сайту повинен бути дозволений тільки адміністраторам.

Деякі сервери після аутентифікації зберігають в cookie або прихованих полях ідентифікатор «ролі» користувача ПЗ. Якщо розмежування доступу ґрунтується на перевірці цього параметра без верифікації приналежності до

ролі при кожному запиті, зловмисник може підвищити свої привілеї, модифікувавши значення cookie.

Відсутність тайм-ауту сесії

У разі якщо для ідентифікатора сесії або облікових даних не передбачений таймаут або його значення дуже велике, зловмисник може скористатися старимиданими для авторизації. Це підвищує уразливість сервера для атак, пов'язаних з крадіжкою ідентифікаційних даних.

Фіксація сесії

Використовуючи цю атаку, зловмисник вказує ідентифікатору сесії користувача задане значення. Залежно від функціональних можливостей сервера, існує декілька способів зафіксувати значення ідентифікатора сесії. Для цього можуть використовуватися атаки типу «міжсайтове виконання сценаріїв», підготовка сайту за допомогою попереднього HTTP-запиту, або інші види атак. Після фіксації значення ідентифікатора сесії, зловмисник чекає моменту, коли користувач увійде до системи [7].

Після входу користувача зловмисник використовує ідентифікатор сесії для отримання доступу до системи від імені користувача.

1.4.5 Атаки направлені на розголошення інформації

Головною ціллю атак даного виду є отримання додаткової інформації про ПЗ WEB-додатку. Завдяки цим вразливостям, зловмисник може визначити ПЗ, що використовується, версії клієнта та серверів, шлях розташування тимчасових файлів, резервних копій.

Індексування директорій

Наданням списку файлів в директорії є нормальною поведінкою WEB-сервера, якщо сторінка, що відображується за умовчанням (index.html/home.html/default.htm), відсутня. Коли користувач запрошує основну сторінку WEB-сайту, він зазвичай вказує доменне ім'я сервера без імені конкретного файлу. Сервер переглядає основну директорію, знаходить в ній файл, використовуваний за умовчанням, і на його основі генерує відповідь. Якщо такий файл відсутній, як відповідь може повернутися список файлів в

директорії сервера. Ця ситуація аналогічна виконанню команди «ls» (Unix) або «dir» (Windows) на сервері і форматуванню результатів у вигляді HTML. В цьому випадку зловмисник може дістати доступ до даних, не призначених для вільного доступу.

Використовуючи індексування директорій, можна дістати доступ до наступних даних:

– резервні копії (.bak, .old, .orig);

– тимчасові файли. Такі файли повинні видалятися сервером автоматично, але іноді залишаються доступними;

– приховані файли, назва яких починається з символу «.»;

– угода про імена. Ця інформація може допомогти передбачити імена файлів або директорій (admin або Admin, back - up або backup);

– перелік користувачів сервера. Дуже часто для кожного з користувачів створюється директорія з ім'ям, заснованим на назві облікового запису;

– імена файлів конфігурації (.conf, .cfg, .config);

– вміст серверних сценаріїв або виконуваних файлів у разі невірно вказаних розширень або дозволів [7].

Існує три основні сценарії отримання списку файлів WEB-сервера:

1 Помилки конфігурації. Подібні проблеми виникають, коли адміністратор помилково вказує в конфігурації сервера цю опцію. Подібні ситуації часто виникають при налаштуванні складних конфігурацій, де деякі директорії мають бути доступні для перегляду;

2 Деякі компоненти WEB-сервера дозволяють отримувати список файлів, навіть якщо це не дозволено в конфігураційних файлах. Зазвичай це виникає в результаті помилок реалізації, коли сервер генерує список файлів при отриманні певного запиту;

3 Бази цих пошукових машин (Google, Wayback machine) можуть містити кеш старих варіантів сервера, включаючи списки файлів.

Ідентифікація програмного забезпечення

Визначення версій програмного забезпечення використовується зловмисником для отримання інформації про використовуваних сервером і клієнтом операційних системах, WEB-серверах та Інтернет-броузерах. Також ця атака може бути спрямована на інші компоненти програмного WEB-забезпечення, наприклад службу каталогу або сервер баз даних або використовувані технології програмування. Зазвичай подібні атаки здійснюються шляхом аналізу різної інформації, що надається WEB-сервером.

Для визначення версій клієнтського програмного забезпечення зазвичай використовується аналіз HTTP-запитів (порядок дотримання заголовків, значення User-agent і так далі). Проте для цих цілей може застосовуватися і інша техніка. Так, наприклад, аналіз заголовків поштових повідомлень, створених за допомогою клієнта Microsoft Outlook, дозволяє визначити версію встановленого на комп'ютері Інтернет-браузеру Internet Explorer.

Наявність детальної і точної інформації про використовувані програмного забезпечення дуже важлива для зловмисника, оскільки реалізція багатьох атак (наприклад переповнювання буфера) специфічно для кожного варіанту операційної системи або програмного забезпечення. Крім того, детальна інформація про інфраструктуру дозволяє понизити кількість помилок.

Просочування інформації

Ці вразливості виникають в ситуаціях, коли сервер публікує важливу інформацію, наприклад, коментарі розробників або повідомлення про помилки, яка може бути використана для компрометації системи. Цінні з точки зору зловмисника дані можуть міститися в коментаріях HTML, повідомленнях про помилки або просто бути присутнім у відкритому вигляді. Існує величезна кількість ситуацій, в яких може статися просочування інформації. Вона не обов'язково призводить до виникнення вразливості, але часто дає зловмиснику інформацію до подальшої побудови атаки. З просочуванням важливої інформації можуть виникати ризики різної міри,

тому необхідно мінімізувати кількість службової інформації, доступної на клієнтській стороні.

Аналіз доступної інформації дозволяє зловмисникові провадити розвідку і отримати уявлення про структуру директорій сервера, використовуваних SQL- запитах, назвах ключових процесів і програм сервера. Часто розробники залишають коментарі в HTML-сторінках і коді сценаріїв для полегшення пошуку помилок і підтримки програмного забезпечення. Ця інформація може варіюватися від простих описів деталей функціонування програми до (у гірших випадках) імен користувачів і паролів, використовуваних при відладці. Просочування інформації може відноситися і до конфіденційних даних, оброблюваним сервером. Це можуть бути ідентифікатори користувача (ІНН, номери водійських посвідчень, паспортів і т.д.), а також поточна інформація (баланс особового рахунку або історія платежів). Багато атак цієї категорії виходять за рамки захисту програмного WEB-забезпечення і переходять в область фізичної безпеки. Просочування інформації в цьому випадку часто виникає коли, в Інтернет-броузері відображується інформація, яка не повинна виводитися у відкритому виді навіть користувачеві [7].

Зворотний шлях в директоріях

Ця техніка атак спрямована на отримання доступу до файлів, директорій і команд, що знаходяться поза основною директорією WEB-сервера. Зловмисник може маніпулювати параметрами URL з метою отримати доступ до файлів або виконати команди, що розташовуються у файловій системі WEB-сервера. Для подібних атак потенційно уразливий будь-який пристрій, що має WEB - інтерфейс. Багато WEB-серверів обмежують доступ користувача певною частиною файлової системи, зазвичай званої WEB document root або CGI root. Ці директорії містять файли, призначені для користувача, і програми, необхідні для отримання доступу до функцій WEB-забезпечення. Більшість базових атак, що експлуатують зворотний шлях, засновані на впровадженні в URL символів «./» для того щоб змінити

розташування ресурсу, який оброблятиметься сервером. Оскільки більшість WEB-серверів фільтрують цю послідовність, зловмисник може скористатися альтернативними кодуваннями для представлення символів переходу по директоріях. Популярні прийоми включають використання альтернативних кодувань, наприклад Unicode («.%u2216» або «.%c0%af»), використання зворотного слешу («.\») в Windows-серверах, символів URLEncode («%2e% 2e% 2f») або подвійного кодування URLEncode («.%255c»). Навіть якщо WEB-сервер обмежує доступ до файлів певним каталогом, ця уразливість може виникати в сценаріях або

CGI-програмах. Можливість використання зворотного шляху в каталогах досить часто виникає в додатках, що використовують механізми шаблонів чи завантажують текст їх сторінок з файлів на сервері. У цьому варіанті атаки зловмисник модифікує ім'я файлу, що передається як параметр CGI - програми або серверного сценарію. В результаті зловмисник може отримати початковий код сценаріїв. Досить часто до імені файлу, що запитується, додаються спеціальні символи - такі, як «%00» - з метою обходу фільтрів.

Передбачуване розташування ресурсів

Передбачуване розташування ресурсів дозволяє зловмисникові дістати доступ до прихованих даних або функціональних можливостей. Шляхом підбору зловмисник може дістати доступ до вмісту, не призначеного для публічного перегляду. Тимчасові файли, файли резервних копій, файли конфігурації або стандартні приклади часто є метою подібних атак. В більшості випадків перебір може бути оптимізований шляхом використання стандартної угоди про імена файлів і директорій сервера. Отримувані зловмисником файли можуть містити інформацію про дизайн додатка, інформацію з баз даних, імена машин або паролі, шляхи до директорій. Приховані файли також можуть містити вразливості, відсутні в основному застосуванні.

1.4.6 Логічні атаки

Атаки цього класу спрямовані на експлуатацію функцій ПЗ або логіки його функціонування. Логіка ПЗ є очікуваним процесом функціонування програми при виконанні певних дій. Як приклади можна привести відновлення паролів, реєстрацію облікових записів, аукціонні торги, транзакції в системах електронної комерції. ПЗ може вимагати від користувача коректного виконання декількох послідовних дій для отримання певного результату. Зловмисник може обійти ці механізми або використовувати їх у своїх цілях.

Зловживання функціональними можливостями

Ця атака спрямована на використання функцій програмного WEB-забезпечення з метою обходу механізмів розмежування доступу. Деякі механізми програмного WEB-забезпечення включаючи функції забезпечення безпеки можуть бути використані для цих цілей. Наявність вразливості в одному з другорядних компонентів ПЗ може привести до компрометації усього ПЗ. Рівень ризику і потенційні можливості зловмисника у разі проведення атаки дуже сильно залежать від конкретного ПЗ. Зловживання функціональними можливостями дуже часто використовується спільно з іншими атаками - такими, як зворотний шлях в директоріях і так далі. Приклади зловживання функціональними можливостями включають:

– використання функцій пошуку для отримання доступу до файлів за межами кореневої директорії WEB-сервера;

– використання функції завантаження файлів на сервер для перезапису файлів конфігурації або впровадження серверних сценаріїв;

– реалізацію відмови в обслуговуванні шляхом використання функції блокування облікового запису при багаторазовому введенні неправильного пароля.

Відмова в обслуговуванні

Цей клас атак спрямований на порушення доступності WEB-сервера. Атаки, спрямовані на відмову в обслуговуванні, реалізуються на мережевому рівні, проте вони можуть бути спрямовані і на прикладний рівень.

Використовуючи функції програмного WEB забезпечення зловмисник може вичерпати критичні ресурси системи або скористатися вразливістю, що призводить до припинення функціонування системи. Зазвичай DoS-атаки спрямовані на вичерпання критичних системних ресурсів - таких, як обчислювальні потужності, оперативна пам'ять, дисковий простір або пропускна спроможність каналів зв'язку. Якщо якийсь з ресурсів досягне максимального завантаження, ПЗ цілком буде недоступне. Атаки можуть бути спрямовані на будь-який з компонентів WEB-забезпечення, наприклад, такі як сервер СКБД, сервер аутентифікації і т.д. [7].

Недостатня протидія автоматизації

Недостатня протидія автоматизації виникає, коли сервер дозволяє автоматично виконувати операції, які повинні проводитися вручну. Для деяких функцій програмного забезпечення необхідно реалізовувати захист від автоматичних атак. Автоматизовані програми можуть варіюватися від нешкідливих роботів пошукових систем до систем автоматизованого пошуку вразливостей і реєстрації облікових записів. Подібні роботи генерують тисячі запитів в хвилину, що може привести до падіння продуктивності усього WEB-серверу. Протидія автоматизації полягає в обмеженні можливостей подібних програмних засобів.

Недостатня перевірка процесу

Вразливості цього класу виникають, коли сервер недостатньо перевіряє послідовність виконання операцій ПЗ. Якщо стан сесії користувача і ПЗ належним чином не контролюється, ПЗ може бути вразливий для шахрайських дій. В процесі доступу до деяких функцій ПЗ очікується, що користувач виконає ряд дій в певному порядку. Якщо деякі дії виконуються невірно або в неправильному порядку, виникає помилка, що призводить до порушення цілісності. Прикладами подібних функцій виступають переведення, відновлення паролів, підтвердження купівлі, створення облікового запису і т.д. В більшості випадків ці процеси складаються з ряду послідовних дій, здійснюваних в чіткому порядку. Для забезпечення коректної роботи подібних

функцій WEB-забезпечення повинно чітко відстежувати стан сесії користувача і її відповідність поточним операціям. В більшості випадків це здійснюється шляхом збереження стану сесії в cookie або прихованому полі форми HTML. Але оскільки ці значення можуть бути модифіковані користувачем,обов'язково повинна проводитися перевірка цих значень на сервері. Якщо цього не відбувається, зловмисник дістає можливість обійти послідовність дій, тобто, логіку ПЗ.

1.5 Протокол HTTP 2.0

HTTP (HyperText Transfer Protocol - протокол передачі гіпертексту) - символьно-орієнтований клієнт-серверний протокол прикладного рівня без збереження стану, який використовується сервісом World Wide Web. Основним об'єктом маніпуляції в HTTP є ресурс, на який вказує URI *(Uniform Resource Identifier* - унікальний ідентифікатор ресурсу) в запиті клієнта. Основними ресурсами що зберігаються на сервері є файли, але ними можуть бути й інші логічні (напр. каталог на сервері) або абстрактні об'єкти (напр. ISBN). Протокол HTTP дозволяє вказати спосіб подання (кодування) одного і того ж ресурсу за різними параметрами: mime-типу, мови і т. д. Завдяки цій можливості клієнт і веб-сервер можуть обмінюватися двійковими даними, хоча даний протокол є текстовим [11].

1.5.1 Структура протоколу

Структура протоколу визначає, що кожне HTTP-повідомлення складається з трьох частин (рис. 1.7), які передаються в наступному порядку:

1 Стартовий рядок (англ. Starting line) – визначає тип повідомлення;

2 Заголовки (англ.) – характеризують тіло повідомлення, параметри передачі та інші відомості;

3 Тіло повідомлення (англ.) – безпосередньо дані повідомлення. Обов’язково повинно відділятися від заголовків порожнім рядком.

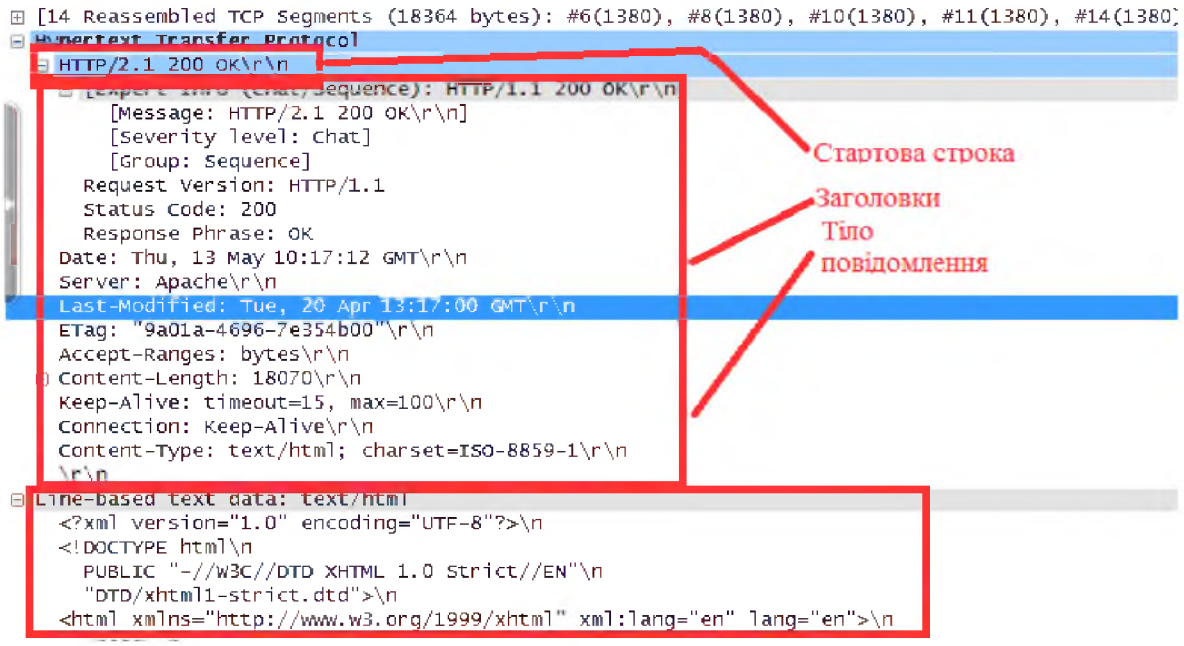


Рисунок 1.7 - Структура протоколу HTTP (Дамп пакета, отриманий сніффером Wireshark)

1.5.2 Стартовий рядок HTTP

Стартовий рядок є обов'язковим елементом, тому що вказує на тип запиту / відповіді; заголовки і тіло повідомлення можуть бути відсутні.

Стартові рядки розрізняються для запиту і відповіді. Рядок запиту виглядає так:

Метод URL HTTP/ Версія протоколу

Приклад запиту:

GET /web-programming/index.html HTTP 1.1

Стартовий рядок відповіді сервера має наступний формат:

HTTP /Версія Код Стану (Пояснення)

Наприклад, на попередній наш запит клієнтом даної сторінки сервер відповів рядком:

HTTP/2.1 200 Ok

1.5.3 Методи протоколу

Метод HTTP (англ. HTTP Method) - послідовність з будь-яких символів, крім керівних і роздільників, яка вказує на основну операцію над

ресурсом. Зазвичай метод являє собою короткий англійське слово, записане заголовними буквами (Табл. 1.1). Назви методу чутливі до регістру [11].

Таблиця 1.1 - Методи протоколу HTTP

Метод	Коротке описання
OPTIONS	Використовується для визначення можливостей веб-сервера або параметрів з'єднання для конкретного ресурсу. Передбачається, що запит клієнта може містити тіло повідомлення для вказівки відомостей що його цікавлять. Формат тіла і порядок роботи з ним у даний момент не визначений. Сервер поки повинен його ігнорувати. Аналогічна ситуація і з тілом у відповіді сервера. Для того щоб дізнатися можливості всього сервера, клієнт повинен вказати в URI зірочку - «*». Запити «OPTIONS * HTTP/2.0» можуть також застосовуватися для перевірки працездатності сервера (аналогічно «пінгування») і тестування на предмет підтримки сервером протоколу HTTP версії 2.0.Результат виконання цього методу не кешується.

Продовження таблиці 1.1

Метод	Коротке описання
GET	Використовується для запиту вмісту зазначеного ресурсу. За допомогою методу GET можна також розпочати будь-який процес. В цьому випадку в тіло відповідного повідомлення слід включити інформацію про хід виконання процесу. Клієнт може передавати параметри виконання запиту в URI цільового ресурсу після символу «?»: GET / path / resource? Param1 = value1 & m2 = value2 HTTP/2.0 Відповідно до стандарту HTTP, запити типу GET вважаються ідемпотентними - багаторазове повторення одного і того ж запиту GET повинне приводити до однакових результатів (за умови, що сам ресурс не змінився за час між запитами). Це дозволяє кешувати відповіді на запити GET. Крім звичайного методу GET, розрізняють ще умовний GET і частковий GET. Умовні запити GET містять заголовки If-Modified-Since, If-Match, If-Range і подібні. Часткові GET містять в запиті Range. Порядок виконання подібних запитів визначено стандартами окремо.
HEAD	Аналогічний методу GET, за винятком того, що у відповіді сервера відсутнє тіло. Запит HEAD звичайно застосовується для вилучення метаданих, перевірки наявності ресурсу (валідація URL) і щоб дізнатися, чи не змінився він з моменту останнього звернення. Заголовки відповіді можуть кешуватися. При розбіжності метаданих ресурсу з відповідною інформацією в кеші копія ресурсу позначається як застаріла.
POST	Застосовується для передачі даних користувача заданому ресурсу. Наприклад, в блогах відвідувачі зазвичай можуть вводити свої коментарі до записів в HTML-форму, після чого вони передаються серверу методом POST і він поміщає їх на сторінку. При цьому передані дані (у прикладі з блогами - текст коментаря) включаються в тіло запиту. Аналогічно за допомогою методу POST звичайно завантажуються файли.

Продовження таблиці 1.1

Метод	Коротке описання
	На відміну від методу GET, метод POST не вважається ідемпотентним, тобто багаторазове повторення одних і тих же запитів POST може повертати різні результати (наприклад, після кожної відправки коментаря з'являтиметься одна копія цього коментаря). При результатах виконання 200 (Ok) і 204 (No Content) в тіло відповіді слід включити повідомлення про підсумок виконання запиту. Якщо був створений ресурс, то серверу слід повернути відповідь 201 (Created) із зазначенням URI нового ресурсу в заголовку Location. Повідомлення відповіді сервера на виконання методу POST не кешується.
PUT	Застосовується для завантаження вмісту запиту на вказаний у запиті URI. Якщо по заданому URI не існувало ресурсу, то сервер створює його і повертає статус 201 (Created). Якщо ж було змінено ресурс, то сервер повертає 200 (Ok) або 204 (No Content). Сервер не повинен ігнорувати некоректні заголовки Content-* що передаються клієнтом разом з повідомленням. Якщо якийсь з цих заголовків не може бути розпізнаний або не допустимий при поточних умовах, то необхідно повернути код помилки 501 (Not Implemented). Фундаментальна відмінність методів POST і PUT полягає в розумінні призначень URI ресурсів. Метод POST припускає, що за вказаною URI буде проводитися обробка переданого клієнтом вмісту. Використовуючи PUT, клієнт припускає, що вміст що завантажується відповідає ресурсу, що знаходиться за даним URI. Повідомлення відповідей сервера на метод PUT не кешуются.
PATCH	Аналогічно PUT, але застосовується тільки до фрагмента ресурсу.
DELETE	Видаляє вказаний ресурс.
TRACE	Повертає отриманий запит так, що клієнт може побачити, що проміжні сервера додають або змінюють в запиті.

Кожен сервер зобов'язаний підтримувати як мінімум методи GET і HEAD. Якщо сервер не розпізнав зазначений клієнтом метод, то він повинен повернути статус 501 (Not Implemented). Якщо серверу метод відомий, але він не застосовний до конкретного ресурсу, то повертається повідомлення з кодом 405 (Method Not Allowed). В обох випадках серверу слід включити в повідомлення відповіді заголовок Allow зі списком підтримуваних методів.

Найбільш затребуваними є методи GET і POST - на людино-орієнтованих ресурсах, POST - роботами пошукових машин і оффлайн-браузерами.

1.5.4 Коди стану

Код стану інформує клієнта про результати виконання запиту і визначає його подальшу поведінку. Набір кодів стану є стандартом, і всі вони описані у відповідних документах RFC.

Кожен код представляється цілим тризначним числом. Перша цифра вказує на клас стану, наступні - порядковий номер стану (рисунок 1.8). За кодом відповіді зазвичай слід короткий опис англійською мовою.

Рисунок 1.8 - Структура коду стану HTTP

Введення нових кодів повинне проводитися тільки після погодження з IETF.

Застосовувані в даний час класи кодів стану та деякі приклади відповідей сервера наведено в табл. 1.2:

Таблиця 1.2 - Коди стану протоколу HTTP

Клас кодів	Коротке описання
1xx Informational(Інформаційний)	У цей клас виділені коди, що інформують про процес передачі. В HTTP/2.0 повідомлення з такими кодами повинні ігноруватися. В HTTP/2.0 клієнт повинен бути готовий прийняти цей клас повідомлень як звичайну відповідь, але нічого відправляти серверу не потрібно. Самі повідомлення від сервера містять тільки стартовий рядок відповіді і, якщо потрібно, декілька специфічних для відповіді полів заголовка. Проксі-сервера подібні повідомлення повинні відправляти далі від сервера до клієнта. Приклади відповідей сервера: 100 Continue (Продовжувати) 101 Switching Protocols (Перемикання протоколів) 102 Processing (Йде обробка)
2xx Success(Успішно)	Повідомлення даного класу інформують про випадки успішного приймання та обробки запиту клієнта. В залежності від статусу сервер може ще передати заголовки і тіло повідомлення. Приклади відповідей сервера: 200 OK (Успішно) 201Created (Створено) 202 Accepted (Прийнято) 204 No Content (Немає вмісту) 206 Partial Content (Частковий зміст)
3xx Redirection (Перенаправлення)	Коди статусу класу 3xx повідомляють клієн-тові, що для успішного виконання операції

Продовження таблиці 1.2

Клас кодів	Коротке описання

	потрібно провести наступний запит до іншого URL. У більшості випадків нова адреса вказується у полі Location заголовка. Клієнт в цьому випадку повинен, як правило, зробити автоматичний перехід. Звернімо увагу, що при зверненні до наступного ресурсу можна отримати відповідь з цього ж класу кодів. Може вийти навіть довгий ланцюжок з перенаправлень, які, якщо будуть проводитися автоматично, створять надмірне навантаження на устаткування. Тому розробники протоколу HTTP рекомендують після другої поспіль подібної відповіді обов'язково запитувати підтвердження на перенаправлення у користувача (раніше рекомендувалося після 5-го). За цим стежити зобов'язаний клієнт, так як поточний сервер може перенаправити клієнта на ресурс іншого сервера. Приклади відповідей сервера: 300 Multiple Choices (Множинний вибір) 301 Moved Permanently (Переміщено назавжди) 304 Not Modified (Не змінювалося)
4xx Client Error(Помилка клієнта)	Клас кодів 4xx призначений для вказівки помилок з боку клієнта. При використанні всіх методів, крім HEAD, сервер повинен повернути в тілі повідомлення гіпертекстове пояснення для користувача. Приклади відповідей сервера: 401 Unauthorized (Неавторизовано) 402 Payment Required (Потрібна оплата) 403 Forbidden (Заборонено) 404 Not Found (Не знайдено) 405 Method Not Allowed (Метод не підтримується) 407 Proxy Authentication Required (Потрібна аутентифікація проксі)

Продовження таблиці 1.2

Клас кодів	Коротке описання
5xx Server Error(Помилка сервера)	Коди 5xx виділені під випадки невдалого виконання операції з вини сервера. Для всіх ситуацій, крім використання методу HEAD, сервер повинен включати в тіло повідомлення пояснення, яке клієнт відобразить користувачеві. Приклади відповідей сервера: 500 Internal Server Error (Внутрішня помилка сервера) 502 Bad Gateway (Непрацюючий шлюз) 503 Service Unavailable (Сервіс недоступний) 504 Gateway Timeout Шлюз не відповідає)

1.5.5 Заголовки HTTP

Заголовок HTTP *(HTTP Header)* - це рядок в HTTP-повідомленні, що містить розділену двокрапкою пару виду «параметр-значення». Формат заголовка відповідає загальному формату заголовків текстових мережевих повідомлень ARPA (RFC 822). Як правило, браузер і веб-сервер додають у повідомлення більш ніж по одному заголовку. Заголовки повинні відправлятися раніше тіла повідомлення і відокремлюватися від нього хоча б одним порожнім рядком (CRLF).

Назва параметра має складатися мінімум з одного друкованого символу (ASCII-коди від 33 до 126). Після назви відразу повинен слідувати символ двокрапки. Значення може містити будь-які символи ASCII, крім перекладу рядка (CR, код 10) і повернення каретки (LF, код 13).

Пробільні символи на початку і наприкінці значення обрізаються. Послідовність декількох пробільних символів всередині значення може сприйматися як один пропуск. Регістр символів в назві і значення не має значення (якщо інше не передбачено форматом поля) [11].

Приклад заголовків відповіді сервера:

Server Apache/2.2.3 (CentOS)

Last-Modified: Wed, 09 May 2015 17:13:15 GMT

Content-Type: text/html; charset=UTF-8

Accept-Ranges: bytes

Date: Thu, 23 May 2015 4:04:36 GMT

Content-Length: 2945

Connection: keep-alive

200 OK

Всі HTTP-заголовки поділяються на чотири основні групи:

1. General Headers (Основні заголовки) – повинні додаватися до будь-якого повідомлення клієнта та сервера.

2. Request Headers (Заголовки запиту) – застосовуються тільки в запитах клієнта;

3. Response Headers (Заголовки відповіді) – присутні тільки у відповідях сервера;

4. Entity Headers (Заголовки сутності) – супроводжують кожну сутність повідомлення;

Сутності *(entity,* в перекладах також зустрічається назва "об'єкт") - це корисна інформація, передана в запиті або відповіді. Сутність складається з метаінформації (заголовки) і безпосередньо вмісту (тіло повідомлення).

В окремий клас заголовки суті виділені, щоб не плутати їх з заголовками запиту або заголовками відповіді при передачі множинного вмісту (multipart / *). Заголовки запиту і відповіді, як і основні заголовки, описують все повідомлення в цілому і розміщуються тільки в початковому блоці заголовків, у той час як заголовки суті характеризують вміст кожної частини окремо, розташовуючись безпосередньо перед її тілом.

У таблиці 1.3 наведено короткий опис деяких HTTP-заголовків.

Таблиця 1.3 - Заголовки HTTP

Заголовок	**Група**	**Коротке описання**
Allow	Entity	Список методів, застосовних до запитуваного ресурсу.
Content-Encoding	Entity	Застосовується при необхідності перекодування вмісту (наприклад, gzip / deflated).
Content-Language	Entity	Локалізація вмісту (мова(и))
Content-Length	Entity	Розмір тіла повідомлення (в октетах)
Content-Range	Entity	Діапазон (використовується для підтримки багато поточного завантаження чи дозавантаження)
Content-Type	Entity	Вказує тип вмісту (mime-type, наприклад text / html). Часто включає вказівку на таблицю символів локалі (charset)
Expires	Entity	Дата / час, після якої ресурс вважається застарілим. Використовується проксі-серверами
Last-Modified	Entity	Дата / час останньої модифікації сутності
Cache-Control	General	Визначає директиви управління механізмами кешування. Для проксі-серверів.
Connection	General	Задає параметри, необхідні для конкретного з'єднання.
Date	General	Дата і час формування повідомлення
Pragma	General	Використовується для спеціальних вказівок, які можуть (опціонально) застосовуватися до будь-якого одержувачу по всьому ланцюжку запитів / відповідей (наприклад, pragma: no-cache).
Transfer-Encoding	General	Задає тип перетворення, що застосовується до тіла повідомлення.

Продовження таблиці 1.3

Заголовок	Група	Коротке описання
Via	General	Використовується шлюзами і проксі для відображення проміжних протоколів і вузлів між клієнтом і веб-сервером.
Warning	General	Додаткова інформація про поточний статус, яка не може бути представлена в повідомленні.
Accept	Request	Визначає застосовні типи даних, очікуваних у відповіді.
Accept-Charset	Request	Визначає кодування символів (charset) для даних, очікуваних у відповіді.
Accept-Encoding	Request	Визначає застосовні формати кодування / декодування вмісту (напр, gzip)
Accept-Language	Request	Відповідні мови. Використовується для узгодження передачі.
Authorization	Request	Облікові дані клієнта, що запитує ресурс.
From	Request	Електронна адреса відправника
Host	Request	Ім'я / мережеву адресу [і порт] сервера. Якщо порт не вказаний, використовується 80.
If-Modified-Since	Request	Використовується для виконання умовних методів Якщо запитуваний ресурс змінився, то він передається з сервера, інакше - з кешу.
Max-Forwards	Request	Представляє механізми обмеження кількості шлюзів і проксі при використанні методів TRACE і OPTIONS.
Proxy-Authorization	Request	Використовується при запитах, що проходять через проксі, що вимагають авторизації
Referer	Request	Адреса, з якого виконується запит. Цей заголовок відсутній, якщо перехід виконується з адресного рядка або, наприклад, по посиланню з js-скрипта.

Продовження таблиці 1.3

Заголовок	Група	Коротке описання
User-Agent	Request	Інформація про користувача агента (клієнта)
Location	Response	Адреса перенаправлення
Proxy-Authenticate	Response	Повідомлення про статус з кодом 407.
Server	Response	Інформація про програмне забезпечення сервера, що відповідає на запит (це може бути як веб-так і проксі-сервер)

1.5.6. Тіло повідомлення

Тіло HTTP повідомлення (*message-body*), якщо воно присутнє, використовується для передачі сутності, зв'язаної із запитом або відповіддю. Приклад тіла повідомлення наведений на рисунку 1.9:

```
<!DOCTYPE html\n
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"\n
  "DTD/xhtml1-strict.dtd">\n
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">\n
  <head>\n
    <title>Ethereal: Download</title>\n
    <style type="text/css" media="all">\n
\t@import url("mm/css/ethereal-3-0.css");\n
    </style>\n
</head>\n
  <body>\n
    <div class="top">\n
    <table width="100%" cellspacing="0" cellpadding="0" border="0" summary="">\n
      <tr>\n
        <td valign="middle" width="1">\n
\t  <a href="/"><img class="logo" title="Ethereal home" src="mm/image/elogo-64-trans.gif" alt="" width="64" height="64"></img></a>\n
        </td>\n
        <td align="left" valign="middle">\n
          <h2>Ethereal</h2>\n
```

Рисунок 1.9 – Приклад тіла повідомлення у HTTP – відповіді.

Тіло повідомлення (*message-body*) відрізняється від тіла сутності (entity-body) тільки у тому випадку, коли при передачі застосовується кодування, вказане у заголовку Transfer-Encoding. У всіх інших випадках тіло повідомлення ідентичне тілу сутності [11].

Заголовок Transfer-Encoding повинен відправлятися для вказівки будь-якого кодування передачі, застосованого додатком у цілях гарантування безпечної та правильної передачі повідомлення. Transfer-Encoding – це

властивість повідомлення, а не сутності, та вона може бути додана або видалена будь-яким додатком у ланцюжку запитів/відповідей [12].

Присутність тіла повідомлення у запиті відмічається додаванням до заголовків запитів поля заголовка Content-Length або Transfer-Encoding. Тіло повідомлення (message-body) може бути додане у запит тільки коли метод запиту допускає тіло сутності (entity-body).

Усі відповіді містять тіло повідомлення, можливо нульової довжини, крім відповідей на запит методом HEAD та відповідей с кодами статусу 1xx (інформаційні), 204 (немає вмісту), та 304 (не модифікований) [11].

1.6 Постановка задачі

Тема захисту WEB-додатків на сьогоднішній час є актуальною – вона постійно розвивається, з'являються нові вразливості. Тому важливим є виділення тих типів вразливостей, які на сьогоднішній день є найбільш розповсюдженими, найменш досліджуваними та тим не менш становлять потенційну загрозу для інформаційної безпеки додатку.

Вразливість типу HTTP Parameter Pollution знайдена дуже недавно, а також експлуатує недоліки протоколу HTTP та HTTPS, які широко використовуються для взаємодії між клієнтом та WEB-додатком. Саме тому задача розробки методики для тестування та виявлення вразливостей цього типу є головним шляхом для вирішення проблеми безпеки WEB-додатків.

Поставлена задачи потребує:

1 Проаналізувати вразливість HTTP Parameter Pollution, причини та передумови її виникнення у додатках.
2 Дослідити особливості реалізації цієї вразливості на стороні клієнту.
3 Розробити методику тестування захищеності WEB-додатків на вразливість типу HTTP Parameter Pollution.
4 Довести практичну цінність даної методики для реального WEB-додатку.

1.7 Висновок

Розглянута класифікація загроз та вразливостей WEB-додатків доводить той факт, що на сьогоднішній час існують проблеми у захисті інформації.

З іншого боку недостатня захищеність найбільш використовуваних протоколів передачі даних в мережі Інтернет - HTTP та HTTPS – представляює собою проблематичну сферу діяльності захисту інформаційних ресурсів різного ступеня складності від несанкціонованого доступу.

Розробка методики, яка дозволить виявити наявність вразливості типу HTTP Parameter Pollution є головною задачею для подолання недостатньої захищеності WEB-додатків.

РОЗДІЛ 2. ВИБІР МЕТОДИКИ ТЕСТУВАННЯ

Об’єкт досліджень – процес тестування захищеності WEB-додатків за допомогою методів «чорного» ящика.

Предмет досліджень – методика тестування захищеності WEB-додатків на вразливість HTTP Parameter Pollution.

Мета НДР – обґрунтувати вибір методики тестування захищеності web – додатків від мережевих атак з можливістю обходу Web Application Firewall та практично довести ефективність запропонованої методики.

Вихідні дані для проведення роботи: державні стандарти України в галузі інформаційної безпеки, нормативні документи з технічного захисту інформації та закони України.

Наукова новизна результатів, що очікуються, полягає у вирішенні проблеми тестування захищеності WEB-додатків від атак зі сторони клієнта на основі використання методики, що включає в себе використання програмних засобів та виконання тестування на проникнення.

Практична цінність результатів полягає у розробленні:

а) методики тестування захищеності WEB-додатків на вразливість HTTP Parameter Pollution, основною перевагою якого є можливість обходу

WAF (Web Application Firewall);

б) аналізу атак на стороні клієнта із використанням вразливості HTTP Parameter Pollution;

Результати повинні відповідати вимогам Закону України «Про вищу освіту», Закону України «Про освіту», «Положення про організацію навчального процесу у вищих навчальних закладах», що затверджено наказом Міністерства Освіти України від 2 червня 1993 р. №161, нормативних документів з технічного захисту інформації, державних стандартів України в

галузі інформаційної безпеки та інших законів України, що стосуються забезпечення безпеки інформації.

Результати досліджень мають бути подано у вигляді, що дозволяє безпосереднє використання методики у комплексі заходів, щодо тестування захищеності.

Економічний ефект повинен бути позитивним завдяки зменшенню витрат на придбання та використання засобів для тестування захищеності WEB-додатків від атак зі сторони клієнта.

Соціальний ефект повинен бути позитивним завдяки розробці методики тестування захищеності WEB-додатків на вразливість HTTP Parameter Pollution, перевагою якого є можливість обходу Firewall .

2.1 Аналіз циркулюючої інформації у WEB-додатках

Всю інформацію, що циркулює у межах функціонування WEB-додатку можна розділити на такі види: клієнтську, серверну та периферійну.

До клієнтської інформації можна віднести вміст форм, що присутні на WEB-сторінці, а також специфіку побудови запиту, який формується на стороні клієнту.

До серверної інформації можна віднести інформацію про тип WEB-серверу, що використовується у додатку; інформацію, що циркулює у базах даних; інформацію щодо механізмів захисту у додатку; інформацію про допоміжні модулі; інформацію про середовище, яке забезпечує зв'язок між основними ресурсами та зовнішніми системами.

Головною особливістю WEB-додатків, що створюються на базі клієнт – серверної архітектури є те, що клієнтські дані обробляються н стороні сервера, що збільшує можливість доступу до даних на WEB-сайті.

До периферійної інформації відносяться дані про особливості WEB-додатка, які можуть бути отримані зі сторони клієнта. До цього типу можна віднести дані про середовище розробки додатку та особливості його функціонування, мову скриптів, що застосовуються у додатку, ПЗ WEB-сервера, ОС та інші елементи.

Ключовою особливістю периферійної інформації є її доступність та відкритість. З позиції зловмисника, дана інформація є цінною для пошуку вразливостей у WEB-додатку, оскільки містить дані про елементи архітектури серверної частини додатку [16].

До такої інформації відносять:

а) HTTP запити та відповіді сервера на них;

б) формат і написи на сторінках, що повідомляють про помилки;

в) вихідні коди доступних сторінок;

Вихідний код сторінок може дати вичерпну інформацію про програму

Приклад:

<title> Home Page </ title>

<meta content="Microsoft Visual Studio 7.0" name="GENERATOR">

<meta content="C#" name="CODE_LANGUAGE">

<meta content="JavaScript" name="vs_defaultClientScript">

У даній ситуації розробник, використовує MS Visual Studio 7. Середовищем є Microsoft IIS 5.0 с. NET framework.

На основі даної інформації можуть бути побудовані власні запити до серверної частини WEB-додатку. Метод тестування захищеності WEB-додатків, який має назву FUZZING – тестування, грунтується на базових знаннях про додаток та основні дані, що супроводжують появлення помилок в роботі додаток.

Для запобігання атакам, які засновані на використанні механізмів автентифікації та авторизації, виконанні коду та інших типів атак, потрібно блокувати доступність периферійної інформації, що циркулює у WEB-додатку. Саме блокування периферійної інформації має бути ключовим напрямком підвищення захисту WEB-додатка.

2.2 Модель загроз

Загрози для WEB-додатку можна класифікувати лише за ймовірністю їх втілення та рівнем впливу на ресурси.

Ймовірність здійснення атаки використовуючи загрози та вразливості:

А – низька;

Б – середня;

В – висока.

Ймовірність та оцінка загроз наведені у таблицях 2.2 та 2.3

Таблиця 2.1 – Модель загроз для WEB-додатків

Загроза	Ймовірність втілення	Рівень впливу
Загрози для операторів		
Атаки на WEB-сервер	В	середній
Навмисне виведення із ладу обладнання	Б	високий
Порушення нормальної роботи: швидкості передачі інформації	Б	високий
Використання службового становища для передачі інформації третім особам	В	високий
Навмисне або не навмисне невірне налагодження обладнання	Б	критичний
Навмисне або не навмисне введені неправдивої інформації	В	високий
Знищення технічних засобів передачі та обробки інформації	А	критичний

Таблиця 2.2 – Модель загроз для користувачів

Загрози для користувачів		
Знищення конфіденційної інформації	Б	середній
Порушення функцій користувача	Б	середній
Блокування профілю користувача	Б	низький
Доступ до інформації користувача	А	високий
Перехоплення даних через додаток	Б	низький
Крадіжка даних, що передаються	В	критичний

Рівні впливу на ресурс:

– критичний – інформація/ресурс/мережа може бути знищена, змінена без можливості відновлення. В даному випадку втрати підприємства є значними;

– високий – інформація/ресурс/мережа може втратити деякі свої властивості, але може бути відновлена. В даному випадку втрати підприємства є меншими, ніж внаслідок повної втрати і неможливості відновити інформацію;

– середній – інформація/ресурс/мережа втрачає деякі властивості, але може бути відновлена в прийнятні терміни і з мінімальними втратами;

– низький – інформація/ресурс/мережа може зазнати невеликих змін, які можливо відновити в найкоротший термін.

2.3 Модель порушника

Модель порушника являє собою формальний опис порушника, його можливих дій та результатів його дій щодо додатка.

По відношенню до АС порушники можуть бути внутрішніми або зовнішніми.

Внутрішніми порушниками можуть бути:

а) адміністратори безпеки WEB-додатків;

б) адміністратори WEB-серверів;

в) адміністратори серверів баз даних.

Зовнішніми порушниками щодо WEB-додатків є усі користувачі, що можуть мати доступ до її клієнтської частини. Зареєстровані користувачі також є зовнішніми порушника, оскільки архітектура будь-якого WEB-додатка регламентує доступність клієнтської частини до інформації, що циркулює на сервері.

Внутрішні порушники мають доступ до всієї циркулюючої інформації на серверній частині, тому результатом їх дій може бути порушення конфіденційності, цілісності та доступності інформації WEB-додатка при будь-яких видах атак чи порушеннях архітектури серверної частини додатка.

Можливість проведення атак зі сторони клієнта, тобто зі сторони зовнішнього порушника є небезпечними за випадку завчасного аналізу WEB-додатка, тобто наявності інформації про архітектуру та опис можливих шляхів

доступу до неї.

Зовнішніми порушниками можуть бути:

а) зареєстровані користувачі додатка;

б) зловмисники, що мають мету порушення доступності, цілісності та конфіденційності інформації, що циркулює в WEB-додатках;

в) аналітики WEB-додатків.

Метою порушника є:

а) отримання інформації у потрібному обсязі та асортименті;

б) наявність можливості вносити зміни в інформаційні потоки у відповідності зі своїми намірами;

в) нанесення збитків шляхом знищення матеріальних та інформаційних цінностей, що є на серверній частині додатків.

Зареєстровані користувачі додатка мають можливість ведення діалогу з АС через клієнтську частину додатка. Мають можливість запуску фіксованого набору завдань за допомогою модулів додатка, що реалізують заздалегідь передбачені функції обробки інформації. Володіють інформацією про основні закономірності формування в ній запитів до сервера даних та вміють користуватися засобами, що їх формують. Використовують при своїх атаках виключно агентурні методи одержання відомостей, тобто роботу с клієнтом.

Зловмисники, що мають мету порушення доступності, цілісності та конфіденційності інформації, що циркулює в додатках можуть створювати і запускати власні програми з новими функціями обробки інформації, що мають на меті атаку на клієнтську частину додатка. Володіють високим рівнем знань та досвідом роботи з технічними засобами системи та їхнього обслуговування та використовують виключно агентурні методи одержання відомостей, тобто ПЗ, що має інструменти щодо досягнення мети порушника.

Аналітики WEB-додатків мають можливість аналізу АС, тобто мати вплив на додаток на рівні сервера, що дозволяє їм розглянути програмне забезпечення системи і конфігурацію її устаткування. Володіють високим рівнем знань у галузі обчислювальної техніки та програмування, проектування

та експлуатації АС. Використовують способи і засоби активного впливу на АС, що змінюють конфігурацію системи, тобто використовують заздалегідь створені моделі та методики, щоб отримати доступ до додатків.

Адміністратори безпеки мають повний обсяг можливостей щодо додатків на рівні проектування, реалізації, впровадження, супроводження АС, аж до включення до складу АС власних засобів з новими функціями обробки інформації. Володіють інформацією про функції та механізм дії засобів захисту. Використовують способи і засоби активного впливу на АС, що змінюють конфігурацію системи, за допомогою доступу до серверної частини додатка та наявності можливості її конфігурувати [14-15].

2.4 Профіль захищеності

Інформація, циркулююча у WEB-додатках повинна мати рівні гарантій, які забезпечують її цілісність, доступність та конфіденційність. Рівні гарантій зіставляються із існуючими профілями захищеності. Рівні гарантій можуть бути розширеними, якщо необхідно виконати необхідні додаткові умови для нормального функціонування додатку згідно запропонованим бізнес – механізмам.

Головними особливостями WEB-додатків є той факт, що забезпечення цілісності та доступності інформації, що циркулює у додатку є найбільш пріоритетним завданням. Висока пріоритетність пояснюється тим, що користувач повинен отримати доступ до потрібної йому інформації у будь – якій момент за допомогою клієнтського програмного забезпечення.

Забезпечення конфіденційності інформації у WEB-додатках включає в себе механізми захисту на серверній стороні із використанням програмних засобів, а також за допомогою процедур, що створюються на етапі розробки додатка.

НД ТЗІ 2.5 – 010 – 03 визначає наступні мінімально необхідні рівні послуг безпеки для забезпечення захисту інформації від загроз:

а) за умови, коли WEB-сервер і робочі станції розміщуються на території установи-власника WEB-сторінки або на території оператора (технологія Т1),

мінімально необхідний функціональний профіль визначається: КА-2, ЦА-1, ЦО-1, ДВ-1, ДР-1, НР-2, НИ-2, НК-1, НО-1, НЦ-1, НТ-1;

б) за умови, коли WEB-сервер розміщується у оператора, а робочі станції – на території власника WEB-сторінки, взаємодія яких з WEB-сервером здійснюється з використанням мереж передачі даних (технологія Т2), мінімально необхідний функціональний профіль визначається: КА-2, КВ-1, ЦА-1, ЦО-1, ЦВ-1, ДВ-1, ДР-1, НР-2, НИ-2, НК-1, НО-1,НЦ-1, НТ-1, НВ-1.

Технологія Т2 є прикладом WEB-додатку, оскільки спосіб передачі інформації від робочої станції, а саме клієнта WEB-додатка до WEB-сервера включає наявність незахищеного середовища тобто мережі Internet, яке не контролюється, і наявністю додаткових вимог щодо ідентифікації та автентифікації між робочої станції й КЗЗ WEB-сервера під час спроби розпочати обмін інформацією та забезпеченням цілісності інформації при обміні.

Таблиця 2.3 – Профіль захищеності WEB-серверів

Критерії	Опис критеріїв
Критерій конфіденційності	
КА-2	Базова адміністративна конфіденційність
КВ-1	Мінімальна конфіденційність при обміні
Критерій цілісності	
ЦА-1	Мінімальна адміністративна цілісність
ЦО-1	Обмежений відкат
ЦВ-1	Мінімальна цілісність при обміні

Продовження таблиці 2.3

Критерії	Опис критеріїв
Критерій доступності	
ДВ-1	Ручне відновлення після збоїв
ДР-1	Використання ресурсів
Критерій спостереженості	
НР-2	Реєстрація
НИ-2	Ідентифікація і автентифікація
НК-1	Однонаправлений достовірний канал
НО-1	Розподіл обов’язків
НЦ-1	Цілісність комплексу засобів захисту
НТ-1	Самотестування за запитом
НВ-1	Автентифікація при обміні

Цей профіль захищеності є базовим для використання та за власником залишається право реалізації, у разі необхідності, окремих послуг безпеки інформації зазначених профілів з більш високим рівнем, доповнення цих профілів іншими послугами, а також реалізація послуг безпеки з більш високим рівнем гарантій [13-15].

2.4.1 Вимоги до реалізації функціональних послуг безпеки інформації WEB-додатка

Базова адміністративна конфіденційність

КЗЗ повинен реалізувати рівень КА-2.

Ця послуга дозволяє адміністратору безпеки керувати потоками інформації від захищених об'єктів до користувачів. Тобто для WEB-додатка цей рівень повинен забезпечувати передачу конфіденційної інформації у вигляді відповіді WEB-сервера до клієнта.

КЗЗ повинен здійснювати розмежування доступу на підставі атрибутів доступу користувача і захищеного об’єкта, тобто на підставі представлення певних ідентифікаторів у запиті зі сторони клієнта, що дозволяють обробляти його на стороні сервера та відповідати користувачу за його ідентифікатором.

Цей механізм можна реалізувати за допомогою механізму сесій, що існує у WEB-додатках після аутентифікації користувача у базі даних.

Конфіденційність при обміні

КЗЗ повинен реалізувати рівень КВ-1.

Ця послуга дозволяє забезпечити захист об'єктів від несанкціонованого ознайомлення з інформацією, що міститься в них, під час їх експорту чи імпорту через незахищене середовище.

КЗЗ повинен забезпечувати захист від безпосереднього ознайомлення з інформацією, що міститься в об'єкті, який передається.

Для забезпечення цього рівня WEB-додаток повинен мати механізми, що шифрують чи захищають запити та відповіді на них від клієнта до сервера і у зворотному напрямку. Такими механізмами є процедури використання протоколів HTTPs, що є модифікацією базового протоку передачі даних від WEB-додатка до клієнта, що використовує шифровані транспортні механізми TSL и SSL.

Мінімальна адміністративна цілісність

КЗЗ повинен реалізувати рівень ЦА-1.

Ця послуга дозволяє керувати потоками інформації від користувачів до захищених об'єктів WEB-сторінки.

КЗЗ повинен здійснювати розмежування доступу на підставі атрибутів доступу користувачів і захищених об'єктів. Розмежування доступу здійснюється на рівні надання користувачеві прав модифікувати об'єкт. Прикладом є той самий механізм сесій з включенням до неї певних додаткових атрибутів.

Права доступу до захищених об'єктів WEB-додатка повинні встановлюватися в момент їх створення або ініціалізації. На етапі розробки потрібно чітко регламентувати до якої інформації клієнтські додатки повинні мати доступ.

Цілісність при обміні

КЗЗ повинен реалізувати рівень ЦВ-1.

Ця послуга дозволяє забезпечити захист WEB-сторінки від несанкціонованої модифікації інформації, яка передається між WEB-сервером

та робочими станціями під час експорту чи імпорту інформації через незахищене середовище. Політика послуги стосується всіх об'єктів, що передаються.

КЗЗ повинен забезпечувати контроль за цілісністю інформації в повідомленнях, які передаються, а також бути здатним виявляти факти їх несанкціонованого видалення або дублювання.

Цілісність при обміні для WEB-додатку забезпечується за допомогою шифрування даних, які циркулюють по каналу зв'язку.

Відкат

КЗЗ повинен реалізувати рівень ЦО-1.

Ця послуга забезпечує можливість відмінити окрему операцію або послідовність операцій і повернути захищений об'єкт після внесення до нього змін до попереднього наперед визначеного стану.

До складу АС, що є системою що забезпечує функціонування WEB-додатка, повинні входити автоматизовані засоби, які дозволяють адміністратору безпеки, користувачу, який має повноваження щодо управління АС, відкатити або відмінити певний набір операцій, виконаних над захищеним об'єктом WEB-сторінки за певний проміжок часу. Тобто для забезпечення цього рівня системи, що забезпечують роботу додатка, повинні мати backup-сервери баз даних, що повинні містити дані для оперативного відновлення нормального функціонування додатка. Цей рівень забезпечує безперервну роботу WEB-додатка, що впливає на його доступність та цілісність.

Використання ресурсів

КЗЗ повинен реалізувати рівень ДР-1.

Ця послуга дозволяє керувати використанням користувачами послуг та ресурсів.

Розмежування доступу користувача до інформації, що циркулює на стороні сервера здійснюється на етапі розробки клієнта, що повинен не мати доступу до серверної частини, окрім запитів з існуючих форм.

Некоректні запити та використання механізму автентифікації у якості забезпечення доступу до цієї інформації повинні реєструватися та блокуватися, тобто WEB-додаток повинен мати механізм перевірки запитів на їх коректність. Забезпечення цього рівня безпеки дозволяє зменшити вразливість від атак класу авторизація.

Цей критерій для WEB-додатка забезпечує його доступність. Для виконання цього критерії можливо використання механізмів конфігурування WEB-серверів. Для WEB-додатків цей критерій забезпечується за конфігурації перезапису mod_rewrite, за допомогою якого сервер модифікує URL при їх завантаженні.

Відновлення після збоїв

КЗЗ повинен реалізувати рівень ДВ-1.

Політика відновлення після збоїв, що реалізується КЗЗ, стосується: системного та функціонального програмного забезпечення; засобів захисту інформації та засобів управління КСЗІ; засобів адміністрування та управління обчислювальною системою АС – і гарантує повернення АС у відомий захищений стан після відмов або переривання обслуговування, спричинених помилковими діями користувачів, неврахованою функціональною недостатністю програмного та апаратного забезпечення (наприклад, можливою наявністю не виявлених під час проектування незадекларованих функцій), іншими непередбачуваними ситуаціями.

Після відмови WEB-сторінки або переривання обслуговування, КЗЗ повинен перевести WEB-сторінку до стану, з якого повернути її в режим нормального функціонування може тільки адміністратор безпеки і користувачі, які мають повноваження щодо управління АС.

Цей рівень для додатків можливо забезпечити за допомогою включення до АС додаткових серверів, що є копіями WEB-серверів та серверів баз даних WEB-додатку, за допомогою яких можна відновити нормальну роботу серверної частини і обробляти запити зі сторони клієнта.

Реєстрація

КЗЗ повинен реалізувати рівень НР-2.

Послуга дозволяє контролювати небезпечні відповідно до політики безпеки WEB-сторінки дії користувачів всіх категорій із захищеними об'єктами.

Критерії реєстрації є базовою потребою для WEB-додатка, оскільки забезпечує конфіденційність роботи клієнта з інформації, яка знаходиться на сервері.

Реєстрація всіх подій, що мають безпосереднє відношення до безпеки, здійснюється в журналі реєстрації, який повинен містити інформацію стосовно дати, часу, місця, типу і наслідків зареєстрованої події, ім'я та ідентифікатор причетного до цієї події користувача. Реєстраційна інформація повинна бути достатньою для однозначної ідентификації користувача, процесу і об'єкта, що мали відношення до кожної зареєстрованої події.

Реєстрація повинна бути реалізована як механізм на серверній частині WEB-додатку у вигляді додаткового модуля. Цим модулем може бути як програмний firewall чи модуль конфігурації сервера, що ідентифікує користувача, а також перевіряє коректність його запитів до сервера.

Ідентифікація і автентифікація

КЗЗ повинен реалізувати рівень НИ-2.

Ідентифікація і автентифікація дозволяють КЗЗ визначити і перевірити особу суб'єкта, що намагається одержати доступ до захищених об'єктів WEB-сторінки. Для WEB-додатку цей критерій забезпечує конфіденційність, оскільки встановлення зв'язку на базі ідентифікатора та зіставлення його з певним клієнтом є механізмом створення безпечного конфіденційного сеансу.

КЗЗ повинен однозначно ідентифікувати категорії користувачів WEB-сторінки і за атрибутами кожної з цих категорій визначати послуги, що їм доступні. Ідентифікація здійснюється на підставі особистого імені та/або IP-адреси користувача.

Дозвіл на виконання будь-яких дій з інформацією та обладнанням WEB-сторінки, що контролюються КЗЗ, надається користувачу тільки після успішного завершення процедур ідентифікації та/або автентифікації його КЗЗ відповідно до категорії користувача.

КЗЗ повинен забезпечувати захист даних автентифікації від несанкціонованого доступу, модифікації або руйнування.

Цей механізм можна реалізувати за допомогою механізму встановлення сесій, а також додаткових механізмів ідентифікації, коли клієнт представляє WEB-додатку певний електронний ключ чи інший елемент, що може встановити особу користувача.

Для сучасних WEB-додатків можливий додатковий модуль для підтвердження свого ідентифікатора з використанням мобільних телефонів, за допомогою якого проходить процедура автентифікації.

Ідентифікація і автентифікація при обміні

КЗЗ повинен реалізувати рівень НВ-1.

Ця послуга дозволяє компонентам КЗЗ WEB-сервера і віддаленої робочої станції здійснити взаємну ідентифікацію, перш ніж розпочати взаємодію.

Обмін інформацією між компонентами КЗЗ повинен здійснюватися тільки після ідентифікації і автентифікації КЗЗ-відправником КЗЗ-отримувача інформації. Результати процедури ідентифікації і автентифікації є дійсними протягом всього сеансу обміну (незалежно від кількості об’єктів, що експортуються) і втрачають свою силу після його закінчення.

Процедура ідентифікації і автентифікація для WEB-додатка та його компонентів КЗЗ повинна здійснюватися на підставі їхніх імен, паролів чи ідентифікаторів встановлених сесій.

Достовірний канал

КЗЗ повинен реалізувати рівень НК-1.

Ця послуга повинна гарантувати користувачу будь-якої категорії можливість безпосередньої взаємодії з КЗЗ, а також те, що ніяка взаємодія

користувача з АС не може бути модифікованою іншим користувачем або процесом. Послуга визначає вимоги до механізму встановлення достовірного зв'язку між користувачем і КЗЗ.

Достовірний канал повинен використовуватися для початкової ідентифікації і автентифікації, тобто створювати цілісний та доступний канал для встановлення сеансу між клієнтом та сервером. Зв'язок з використанням даного каналу повинен ініціюватися виключно користувачем.

Для забезпечення цього рівня WEB-додаток повинен мати механізми, що шифрують чи захищають запити та відповіді на них від клієнта до сервера і у зворотньому напрямку. Такими механізмами є процедури використання протоколів HTTPS, що є модифікацією базового протоку передачі даних від WEB-додатка до клієнта, що використовує шифровані транспортні механізми TSL и SSL.

Розподіл обов'язків

КЗЗ повинен реалізувати рівень НО-1.

Ця послуга дозволяє розмежувати повноваження користувачів, визначивши категорії користувачів з певними і притаманними для кожної з категорій функціями.

КЗЗ повинен присвоїти користувачу атрибути, якими однозначно характеризується надана йому роль. Користувач може виступати в певній ролі тільки після того, як він виконає дії, що підтверджують прийняття ним цієї ролі. Цей механізм додатку допомагає клієнту мати певний рівень доступності до нього та створюється на етапі реєстрації користувача додатка, коли до бази даних користувач потрапляє з певними повноваженнями чи обмеженнями користуванням WEB-додатком, інформацією, що циркулює у ньому чи інших компонентів серверної частини додатка.

Цілісність комплексу засобів захисту

КЗЗ повинен реалізувати рівень НЦ-1.

Ця послуга визначає міру здатності КЗЗ WEB-сторінки захищати себе і гарантувати свою спроможність керувати захищеними об'єктами.

КЗЗ повинен повідомляти адміністратора безпеки про порушення цілісності будь-якого компонента КЗЗ. WEB-сторінка під час цього має бути переведена до стану, в якому доступ до неї користувачів, крім адміністратора безпеки, заборонено.

Компоненти КЗЗ WEB-додатку повинні у разі виключної ситуації звертатися до еталонних компонент, що дозволяють нормально функціонувати серверній частині. Це реалізується за допомогою використання серверів, де знаходяться копії додатка, які є еталонними.

Самотестування

КЗЗ повинен реалізувати рівень НТ-1.

Самотестування дозволяє КЗЗ перевірити і на підставі цього гарантувати правильність функціонування і цілісність певної множини функцій захисту WEB-сторінки.

КЗЗ повинен забезпечувати відповідність набору тестів (неможливість будь-якої модифікації) версії КЗЗ. Зміна тестів можлива лише у процесі інсталяції нової версії КЗЗ.

Наявність тестів у вигляді програмного забезпечення дозволяє створювати ефективний КЗЗ, оскільки тести дозволяють знайти вразливості додатка та за допомогою власних механізмів закрити напрями атак на серверну частину з боку клієнта.

Розробка методики тестування захищеності WEB-додатка і є найбільш значимою. Кожний з WEB-серверів та додатків використовує лише тестування системи на предмет інтеграції до них вірусів чи інших елементів, що можуть зупинити роботу АС.

2.4.2 Вимоги до реалізації критеріїв гарантій WEB-сторінки

Критерії гарантій включають вимоги до архітектури КЗЗ, середовища розробки, послідовності розробки, середовища функціонування, документації, випробувань КЗЗ.

Гарантії реалізації послуг безпеки повинні відповідати рівню Г2 у відповідності до НД ТЗІ 2.5-004.

Архітектура

Програмне забезпечення, призначене для реалізації КЗЗ, повинне будуватися за модульним принципом.

Склад послуг безпеки, а також механізмів захисту, що реалізують кожну з послуг, визначається політикою безпеки інформації в АС і повинен відповідати її вимогам. Якщо не всі вимоги політики безпеки реалізуються КЗЗ, то вони повинні підтримуватися організаційними та іншими заходами захисту КСЗІ. У складі КЗЗ не повинні міститися послуги та використовуватися засоби, які мають не передбачені політикою безпеки функції. Використання таких засобів можливе за умови вилучення цих функцій або гарантування неможливості їх активізації.

Мають бути описані особливості архітектури компонентів КСЗІ та їх призначення. Стиль опису – неформалізований, вимоги щодо детального опису не висуваються.

Середовище розробки

Мають бути визначені всі стадії та етапи життєвого циклу АС, а для кожної стадії та етапу – перелік і обсяги необхідних робіт та порядок їх виконання. Всі стадії та етапи робіт повинні бути задокументовані. Види та зміст документів встановлено державними стандартами.

На всіх стадіях життєвого циклу повинні існувати процедури керування конфігурацією АС. Ці процедури повинні визначати технологію відслідковування та внесення змін в апаратне та програмне забезпечення КСЗІ, тестове покриття і документацію та гарантувати, що без дотримання цієї технології ніякі зміни не можуть бути внесені. Технологія слідкування та внесення змін повинна гарантувати постійну відповідність між документацією і реалізацією поточної версії КЗЗ.

Послідовність розробки

Для всіх стадій життєвого циклу АС повинні бути розроблені функціональні специфікації КСЗІ.

На підготовчому етапі створення КСЗІ має бути виконане обстеження середовищ функціонування АС, в результаті якого визначаються об'єкти захисту, здійснюється класифікація інформації та розробляється модель загроз для інформації і концепція політики безпеки інформації в АС. На підставі цих даних мають бути сформульовані функціональні специфікації вимог із захисту інформації в АС. Ці специфікації мають бути викладені в окремому розділі в технічному завданні на створення АС або окремому технічному завданні на створення КСЗІ.

Функціональні специфікації політики безпеки і моделі політики безпеки повинні містити перелік і опис послуг безпеки, що надаються КЗЗ, а також правила розмежування доступу до захищених об'єктів АС.

Функціональні специфікації проекту архітектури КСЗІ повинні містити модель захисту, де враховані всі суттєві загрози і для кожної з них визначено можливі варіанти їх блокування (попередження) за допомогою КЗЗ або організаційними чи іншими заходами захисту. Якщо існує неоднозначність, повинні надаватися додаткові аргументи на користь вибору того чи іншого варіанту.

Функціональні специфікації детального проекту КСЗІ повинні містити принципи побудови, функціональні можливості, опис функціонування кожного механізму захисту та взаємодії механізмів між собою у складі КЗЗ. Повинні бути розроблені документи, що регламентують використання засобів КЗЗ, а також організаційних та інших заходів захисту, які входять до КСЗІ. Як реалізація детального проекту може розглядатися технічний або робочий проекти.

Окремі етапи робіт повинні бути задокументовані відповідно до вимог НД ТЗІ 1.4-001-2000 у вигляді окремих розділів плану захисту інформації в АС або вимог інших нормативно-правових актів і нормативних документів з ТЗІ.

Середовище функціонування

Повинні існувати засоби інсталяції, генерації і запуску КЗЗ, які гарантують, що експлуатація АС починається з безпечного стану, а також існувати документи (інструкції), які регламентують порядок керування цими процедурами. Якщо можливі різні варіанти конфігурації КЗЗ, то всі вони повинні бути описані в інструкціях.

Документація

Документація на КЗЗ у вигляді окремих документів або розділів інших документів повинна містити опис послуг безпеки, що реалізуються КЗЗ, а також настанови для різних категорій користувачів (адміністратора безпеки, адміністратора баз даних, адміністратора сервісів, звичайного користувача тощо) стосовно використання послуг безпеки.

Випробування

Випробування КЗЗ можуть проводитись як самостійно, так і у складі КСЗІ. Для проведення випробувань розробник КЗЗ повинен підготувати програму і методику випробувань, розробити процедури (тести) випробувань усіх механізмів, що реалізують послуги безпеки.

Програма і методика випробувань КЗЗ, тестове покриття, результати випробувань КЗЗ входять до складу обов'язкового комплекту документації, яка надається організатору експертизи під час проведення державної експертизи КСЗІ в АС [14 – 15].

2.5. Аналіз вразливості HTTP Parameter Pollution

2.5.1 Загальні відомості про вразливість

У той час, коли вразливості ін'єкцій для WEB-додатків, такі як SQL - ін'єкції та міжсайтове виконання коду, є добре відомими та вивченими, новому класу вразливостей під назвою HTTP Parameter Pollution (HPP) не приділялося багато уваги. Вразливість HPP вперше була представлена у 2012

році двома спеціалістами з інформаційної безпеки – ді Паоло та Карретоні на конференції OWASP у Польщі [17].

Вразливість HTTP Parameter Pollution (HPP) дозволяє зловмиснику вставляти параметри всередину URL-ів, що генеруються WEB-додатком. Результат цієї атаки залежить насамперед від логіки роботи додатка та може змінюватися від простих незручностей до повної зміни поведінки WEB-додатку.

Наприклад, розглянемо типові HTTP GET та POST – запити.

Приклад HTTP GET – запиту:

GET /foo?par1=val1&par2=val2 HTTP 1.1

User-Agent: Mozilla 39.0

Host: Host

*Accept: */**

Приклад HTTP POST – запиту:

POST /foo HTTP 2.0

User-Agent: Mozilla 39.0

Host: Host

*Accept: */**

Content-Length: 19

par1=val1&par2=val2c

В обох випадках вразливими є параметри запитів - *par1=val1&par2=val2*

Незважаючи на те, що додавання нового параметру може інколи бути достатнім для здійснення атаки на WEB-додаток, зловмисник, як правило, більш зацікавлений у зміненні значення вже існуючого параметру. Це може бути досягнене із використанням так званого «маскування» існуючого параметру – додаванням нового параметру з таким же самим ім'ям [19].

В даному випадку наведений вище HTTP GET – запит буде включати в себе такі параметри:

par1=val1& par1=val3&par2=val2

Яким чином веб – сервер буде обробляти подібний запит залежить насамперед від типу веб – серверу.

2.5.2 Пріоритет параметрів HTTP-GET та HTTP-POST запитів

Для більш повного розуміння вразливості HPP потрібно розглянути поняття пріоритету параметру у HTTP GET та POST – запитах.

Протягом взаємодії із WEB-додатком користувачу часто потрібно ввести деякі дані до програми, що потім згенерує WEB-сторінку, що була потрібна. Протокол HTTP дозволяє браузеру користувача передавати інформацію всередині самого URL (GET параметри), у заголовках HTTP – запитів (поле Cookie) та всередині тіла запиту (POST параметри) . Використання того, чи іншого способу передачі інформації залежить від WEB-додатку, а також від типу та кількості даних, що передаються [20].

Для прикладу, розглянемо WEB-сторінку, що містить такий елемент, як чек бокс, що дозволяє користувачеві вибрати один або декілька варіантів у формі. (рисунок 2.1).

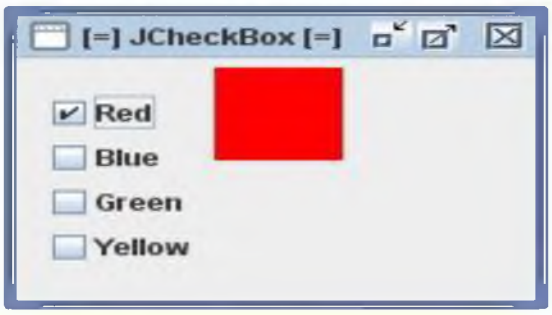


Рисунок 2.1 – Приклад чек бокс елементу на WEB-сторінці

У типовій реалізації, WEB-сторінки, всі пункти чек боксу мають однакове ім’я, і тому в подальшому WEB-браузер буде відправляти окремий параметр для кожного пункту, який був обраний користувачем. Для підтримки цієї функціональності, більшість мов програмування, які використовуються для розробки сторінок, використовують методи, що отримують весь список значень для конкретного параметру [20].

Наприклад JSP має метод:

String[] parameterValues = **request.**getParameterValues();

Цей метод збирає всі значення параметру та повертає список строк.

Але проблема постає, коли розробник очікує прийняти одне значення для одного параметру.

Наприклад:

String variableName = **requset**.getParameter(**"txtUserName"**);

Даний метод повертає лише одне значення. Припустимо, що даному методу в запиті передається декілька параметрів з однаковими іменами та різними значеннями. В такому випадку метод getParameter може повернути перше, останнє або комбінацію значень параметрів, що передаються у запиті.

На даний момент результат подібних запитів залежить від мови програмування, яка застосовується для розробки WEB-додатку та WEB-сервера, який обробляє дані запити [21].

Важливо відзначити той факт, що WEB-додаток повертає тільки одне значення не є вразливістю. Але якщо веб – розробник не зверне увагу на дану проблему, в майбутньому присутність дубльованих параметрів у запитах може спричинити аномальну поведінку WEB-додатку, а також може бути потенційно використане зловмисником у комбінації з іншими видами атак [17].

Існує декілька видів вразливості HTTP Parameter Pollution, але основним видом вважається вразливість на стороні клієнта.

2.5.3 Вразливість HTTP Parameter Pollution на стороні клієнта

Атаки, що використовуються вразливість HTTP Parameter Pollution, можуть бути проведені тоді, коли зловмисний параметр $П_{злов.}$ разом із роздільником строки запиту (наприклад з символом «&») вводиться до існуючого параметру $П_{icн.}$. Якщо параметр $П_{icн}$ не перевіряється належним чином на стороні WEB-додатку та його значення у подальшому кодується та застосовується для генерації URL A, то зловмисник може додати один або декілька нових параметрів до A [17].

Розглянемо типовий сценарій атаки на стороні клієнта. Суть цієї атаки полягає у переконанні жертви перейти по зловмисному URL, що використовує вразливість HPP. Для прикладу, розглянемо WEB-додаток, що дозволяє користувачеві віддати свій голос у різних типах голосувань. WEB-додаток, написаний із застосування технології Java Server Pages (JSP), отримує єдиний параметр, під назвою poll_id, який унікально ідентифікує голосування, в якому користувач бере участь в даний момент. На основі значення цього параметру, WEB-додаток генерує сторінку, що включає в себе список посилань, кожна з яких дозволяє голосувати за одного кандидата.

Наприклад, наступний рисунок (Рисунок 2.2) ілюструє сторінку для голосування, що включає в себе двох кандидатів, на якій користувач може віддати свій голос за того, чи іншого кандидата, натисканням на бажане посилання.:

```
URL: http://host/election.jsp?poll_id=4568
Link 1: <a href="vote.jsp?poll_id=4568&candidate=white"> Vote for mr.White</a>
Link 1: <a href="vote.jsp?poll_id=4568&candidate=green"> Vote for ms.Green</a>
```

Рисунок 2.2 – Код посилань на тестовій сторінці для голосування

Припустимо, що є зловмисник, який зацікавлений у перемозі мс. Грін. Аналізуючи WEB-сторінку, він розуміє, що , WEB-додаток належним чином не перевіряє параметр poll_id. В такому випадку, зловмисник може використати вразливість HPP для додавання іншого параметру до голосування.

Для того, щоб реалізувати атаку, зловмисник генерує модифіковане посилання. Модифіковане посилання наведено на рисунку 2.3:

```
URL: http://host/election.jsp?poll_id=4568%26candidate%3Dgreen
```

Рисунок 2.3 – Модифікований зловмисником запит

Після цього модифіковане посилання відправляється потенційній жертві. Користувач, що переходить по даному зміненому Url буде перенаправлений на оригінальну WEB-сторінку, де зможе віддати свій голос у голосуванні.

Проте, параметр poll_id використовується WEB-додатком для генерації посилання на сторінці. Тому після переходу по модифікованому Url, зловмисне значення *candidate* буде вставлене в усі посилання на сторінці (рисунок 2.4):

```
URL: http://host/election.jsp?poll_id=4568%26candidate%3Dgreen
Link 1: <a href=”vote.jsp?poll_id=4568&candidate=green&candidate=white”> Vote for Mr.White</a>
Link 1: <a href=”vote.jsp?poll_id=4568&candidate=green&candidate=green”> Vote for Mrs.Green</a>
```

Рисунок 2.4 – Сторінка голосування із модифікованими посиланнями

В цьому разі неважливо, на яке посилання натисне користувач - WEB-додаток (в нашому випадку JSP-скрипт vote.jsp) буде отримувати два параметри із назвою *candidate*. Крім того, перший параметр завжди буде мати значення *green*.

Можна зробити припущення, що розробник даного WEB-додатку очікував отримати від користувача тільки одне ім’я кандидата. До того ж, специфіка роботи даного WEB-додатку, написаного із застосуванням технології Java Server Pages, полягає в тому, що коли отримані у запиті параметри обробляються на стороні сервера, тільки перший параметр з іменем *candidate* буде прийнятий. Другий параметр з таким же іменем буде відкинутий.

В даному випадку після модифікації посилання зловмисником, користувач, незалежно від свого вибору, буде голосувати за мс. Грін.

Роблячи висновок, потрібно зазначити, що якщо WEB-додаток є вразливим до HPP, зловмиснику стає можливим модифікувати Url таким

чином, що після одного переходу по даному Url, зміст WEB-сторінки буде змінений – усі посилання будуть вести до голосування за Мс. Грін.

2.6 Методика тестування захищеності WEB-додатків на вразливість HTTP Parameter Pollution

Дана методика представляє собою алгоритм тестування та рекомендації, щодо проведення перевірки захищеності WEB-додатків. Методика базується на детальному вивченні даних про WEB-додаток, про http-запити та http-відповіді від сервера, що супроводжують роботу додатку. Запропонована методика передбачає застосування спеціалізованого програмного забезпечення для модифікації http-запитів та дослідження системи.

Загальний алгоритм тестування згідно запропонованої методики зображений на рисунку 2.5.

Спеціалізоване програмне забезпечення, що використовується у додатку включає в себе:

- Zenmap – сканування додатку;
- Burpsuite – перехоплення та модифікація запиту;
- WebScarab – проксі-сервер для запису модифікованого запиту.

Можливо використовувати інші проксі-сервера (наприклад Paros Proxy) та засоби сканування додатку.

Рисунок 2.5 – Загальний алгоритм тестування захищеності на вразливість HTTP Parameter Pollution

2.6.1 Аналіз WEB – серверу, що використовується у додатку

Для проведення тестування захищеності WEB – додатку насамперед потрібно визначити (або отримати інформацію) щодо WEB – серверу, який обробляє HTTP – запити від користувачів.

В загальному випадку при проведенні тестування можливі два випадки:

1 Вся потрібна інформація щодо WEB – додатку надана замовниками тестування. В цьому випадку потрібно проаналізувати отримані дані та виділити ключові факти, які будуть потрібними при подальшому проведенні тестування.

Насамперед це:

- Тип та версія WEB – серверу, що використовується в даному додатку;
- Технології, що застосовувалися при розробці WEB – додатку;

2 Всієї потрібної для тестування інформації замовниками не надано.

В цьому випадку потрібно застосовувати допоміжне програмне забезпечення – програми Nmap(ZenMap) та BurpSuite.

Розглянемо отримання необхідної інформації на прикладі програми Nmap(ZenMap).

Zenmap 5.61 – утиліта для дослідження мережі та перевірки безпеки. Методика тестування за допомогою Zenmap заснована на використанні IP-пакетів з різними властивостями, які можуть допомогти для визначення доступних хостів у мережі, служб, операційних систем, які обслуговують ці хости. Також дані пакети дозволяють визначити типи пакетних фільтрів або брандмауерів, що допомагає досліджувати механізми захисту додатків.

Вихідними даними Zenmap є список просканованих цілей з додатковою інформацією по кожній в залежності від заданих опцій. Ключовою інформацією є таблиця, де міститься інформація про доступність портів і хостів. Ця таблиця містить номер порту, протокол, ім'я служби і стан. Стан може мати значення open, filtered, closed або unfiltered. Відкрито означає, що додаток на цільовій машині готовий для з'єднання або прийняття пакетів на цей порт. Фільтрується означає, що брандмауер, мережевий фільтр або якась інша перешкода в мережі блокує порт, і утиліта не може встановити відкритий цей порт або закритий. Закриті порти не пов'язані ні з додатками, так що вони можуть бути відкриті в будь-який момент.

Порти розцінюються як не фільтровані, коли вони відповідають на запити, але утиліта не може визначити, відкриті вони або закриті. Zenmap видає комбінації, коли порт відкритий і фільтрується або закритий, але також фільтрується, коли не може визначити, яке з цих двох станів описує порт. Ця таблиця також може надавати деталі про версію програмного забезпечення, якщо це було запитано.

На додаток до таблиці важливих портів утиліта може надавати подальшу інформацію про цілі: перетворені DNS імена, припущення про використовуваної операційній системі, типи пристроїв і MAC адреси.

Zenmap використовує безліч різних методів сканування, таких як UDP, TCP (connect), TCP SYN (напіввідкрите), FTP proxy (прорив через ftp), Reverse-ident, ICMP (ping), FIN, ACK, Xmas tree, SYN та NULL сканування.

Приклад використання програми Zenmap для отримання даних, щодо WEB – серверу, який застосовуються у додатку наведений на рисунку 2.6:

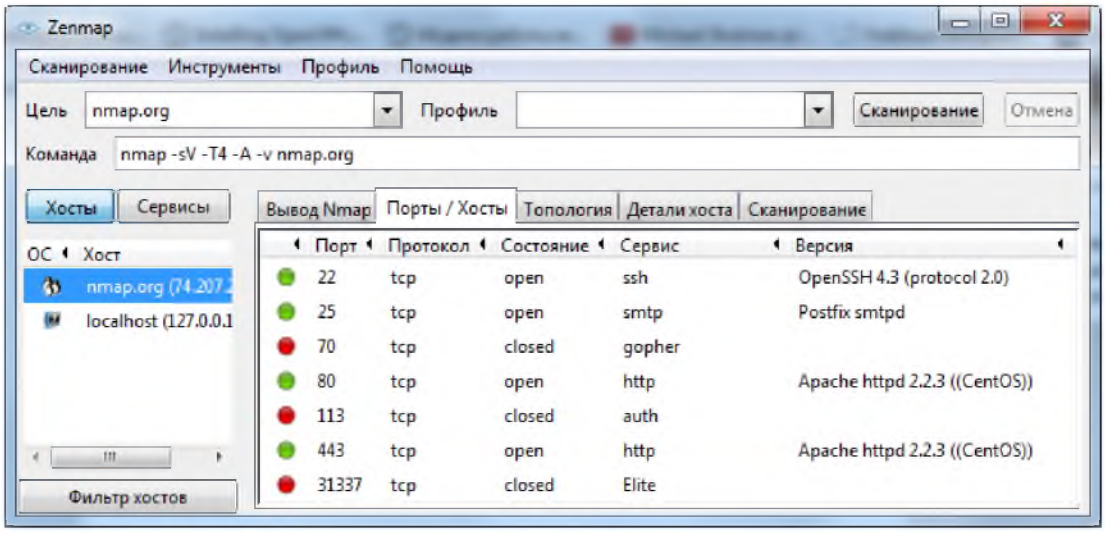


Рисунок 2.6 – Сканування за допомогою програми Zenmap

2.7.2 Визначення способу фільтрації параметрів в HTTP – запитах

Після отримання базової інформації, щодо технології розробки та WEB-серверу, що застосовується у додатку, потрібно визначити яким чином здійснюється фільтрація параметрів у HTTP – запитах.

Для прикладу розглянемо HTTP – запит у якому передаються два

параметри з однаковим ім'ям та різними значеннями:

par1=val1&par1=val2

Перше входження параметру означає, що при прийомі на стороні сервера HTTP – запиту, перший параметр обробляється, а всі параметри, що передаються після нього – відкидаються.

Останнє входження параметру означає, що останній параметр у HTTP – запиті обробляється, а всі інші – відкидаються.

У таблиці 2.5 наведено специфіку фільтрації параметрів у HTTP – запитах в залежності від пріоритетів цих параметрів та технології/ WEB-серверу [22].

Таблиця 2.5 – Пріоритет параметру при передачі декількох параметрів з однаковим ім'ям

Технологія/Сервер	Пріоритет параметрів	Приклад
ASP.NET/IIS	Всі входження параметру	par1=val1,val2
ASP/IIS	Всі входження параметру	par1=val1,val2
PHP/nginx	Останнє входження	par1=val2
JSP,Servlet/nginx	Перше входження	par1=val1
PHP/Apache	Останнє входження	par1=val2
JSP,Servlet/Apache Tomcat	Перше входження	par1=val1
mod_perl,libapreq2/Apache	Перше входження	par1=val1
Perl CGI/Apache	Перше входження	par1=val1
mod_wsgi (Python)/Apache	Всі входження параметру	par1=val1,val2
JSP,Servlet/Oracle Aplication Server 10g	Перше входження	par1=val1

У разі відсутності технології чи WEB – серверу у таблиці, визначення способу фільтрації параметрів може проводитися на кроці 3 даної методики.

2.6.3 Збір додаткових даних про WEB – додаток

Перед тим, як перейти до тестування на вразливість HTTP Parameter Pollution, необхідно отримати додаткову інформацію щодо роботи WEB – додатку.

До цієї інформації належить:

1. Дані про «нормальну» роботу WEB – додатку. Під «нормальною» роботою розуміється всі HTTP – запит та відповіді від сервера, які виникають у процесі користування додатком.
2. Дані щодо основних параметрів HTTP – запитів, що використовуються у процесі роботи додатку.

Для дослідження «нормальної» роботи WEB – додатку слід використовувати додаткове програмне забезпечення.

На рисунку 2.7 зображено принцип дослідження «нормальної» роботи WEB – додатку.

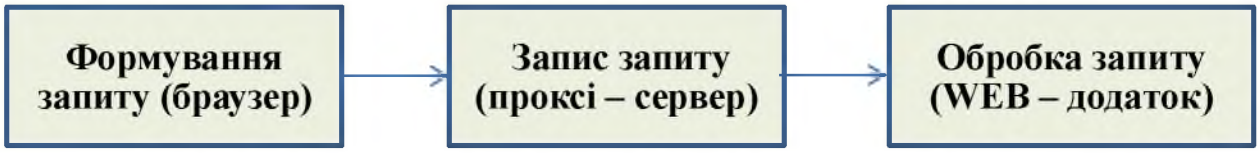


Рисунок 2.7 – Схема дослідження «нормальної» роботи WEB – додатку

1 В якості браузеру використовується один із найпоширеніших на сьогоднішній час браузерів: Mozilla Firefox, Google Chrome, Opera, Internet Explorer, Safari.

2 В якості проксі – серверу може використовуватися як звичайні проксі-сервера (наприклад Paros Proxy), так і спеціалізоване програмне забезпечення для інспектування мережевого трафіку (WebScarab, BurpSuite), які мають функції проксі сервера.

Принцип дослідження згідно схеми, зображеної на рисунку 2.7, полягає у тому, що проводиться тестування з позиції користувача. Виконується перехід по посиланням, інші дії з позиції користувача, а усі HTTP – запити та HTTP – відповіді відображаються на проксі – сервері.

Після проведення дослідження є доступ до переліку HTTP – запитів та відповідей, які вважаються коректними для даного WEB – додатку та не призводять до помилок у його роботі.

Приклад переліку HTTP – запитів отриманого із застосуванням Paros Proxy наведений на рисунку 2.8:

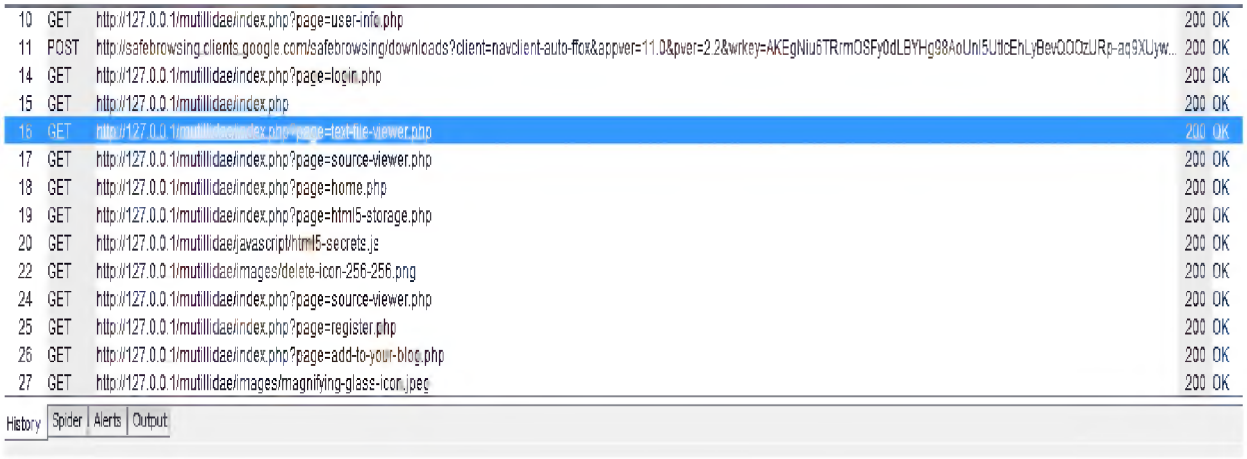


Рисунок 2.8 – Перелік діючих HTTP – запитів у WEB – додатку.(Paros proxy)

Після отримання переліку HTTP – запитів потрібно детально дослідити параметри, що в них передаються.

Особливу увагу слід приділити одиничним параметрам, що передаються у процесі роботи WEB – додатку.

Наприклад:

http://127.0.0.1/mutillidae/index.php?page=userpoll.php&choice=nmap&initials=alexander&user-poll-php-submit-button=Submit+Vote.

У даному запиті передаються такі параметри:

choice=nmap – означає вибір користувача;

initials=alexander – дані про користувача;

Дані параметри слід ретельно дослідити, виконуючи тестування на проникнення.

Проведення тестування на проникнення

Тестування на проникнення є останнім та головним кроком у наведеній методиці. Всі попередні кроки були підготовчими та виконувалися для полегшення проведення тестування захищеності WEB – додатку.

Для проведення тестування на проникнення пропонується наступна схема, зображена на рисунку 2.9:

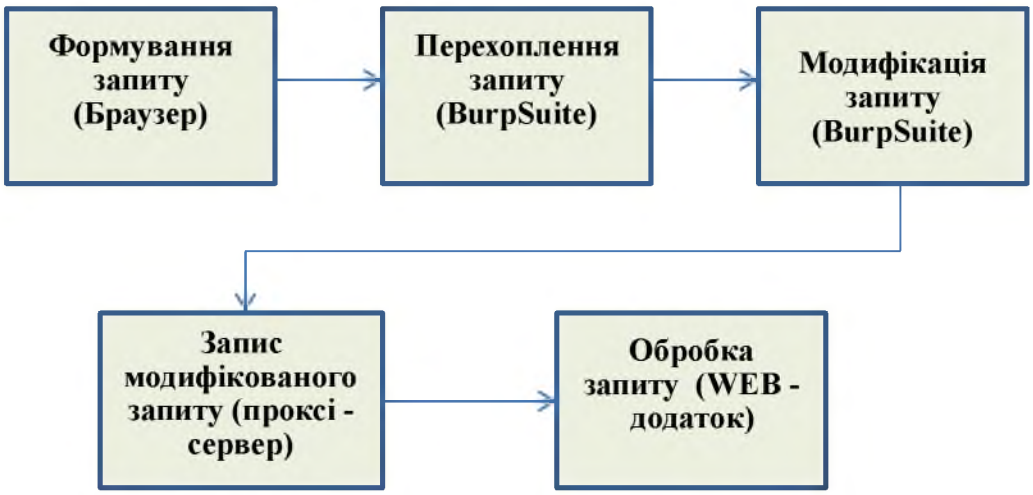


Рисунок 2.9 – Схема тестування WEB – додатку на проникнення

Розглянемо кожен елемент схеми окремо:

1 За допомогою браузеру виконується взаємодія із WEB – додатком. В якості браузеру можуть виступати: Mozilla Firefox, Google Chrome, Opera, Internet Explorer, Safari;

2 Програмний продукт Burpsuite служить головним чином для перехвату та модифікації HTTP – запитів між браузером та WEB – додатком. Burpsuite представляє собою набір відносно незалежних міжплатформенних додатків, написаних на мові Java. Ядром програми є Burp Proxy, який виконує функції локального проксі – сервера; всі останні компоненти програми - це Spider, Intruder, Repeater, Sequencer, Decoder, Comparer.

3 Після перехвату та модифікації HTTP – запиту, Burpsuite перенаправляє його через другий проксі – сервер до веб – додатку.

4 Другий проксі – сервер служить лише для запису HTTP – запитів після їх модифікації у Burpsuite.

Розглянемо процес перехвату та модифікації HTTP – запиту більш детально.

1 Після того, як браузер на стороні клієнта сформував HTTP – запит за допомогою компоненту Burp Proxy виконується перехват даного запиту (рисунок 2.10);

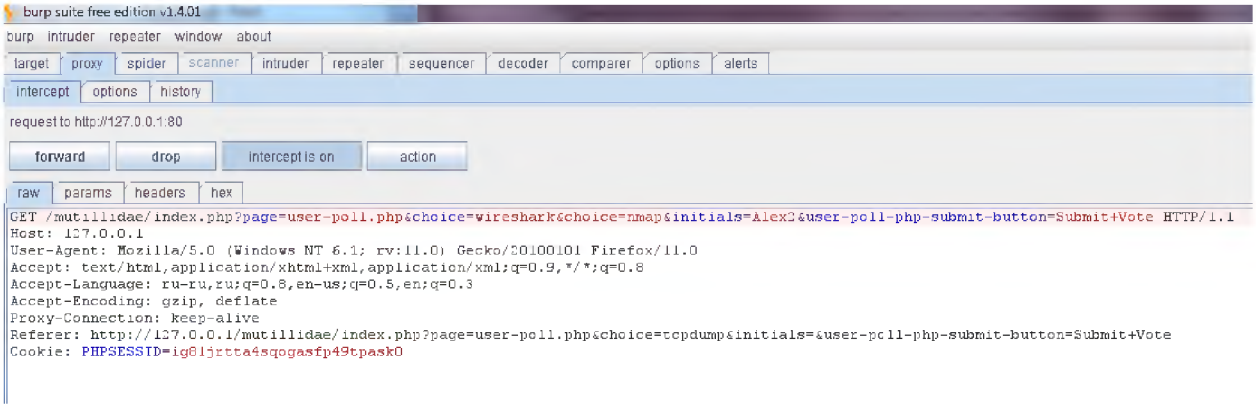


Рисунок 2.10 – HTTP – запит, перехоплений за допомогою компоненту Burp Proxy

2 Після цього потрібно розглянути перехоплений запит та визначити основні параметри, що в ньому передаються, а також символи розділення між параметрами.

У випадку, коли передається одиничний параметр – додається ще один параметр із таким же іменем, але іншим значенням разом із додаванням символу розділення. Потрібно також звернути увагу, яким чином фільтрується параметр на стороні серверу. Якщо фільтрується перше входження параметру, то модифікований параметр слід вставити у запит до основного параметра. Якщо ж навпаки – після основного параметру.

Наприклад перехоплений запит містить параметр par1 та фільтрується перше входження параметру:

*http://host/index.php?**par1=val1***

Тоді запит модифікується наступним чином:

*http://host/index.php?**par1=val2**&par1=val1*

3 Модифікований запит відправляється до WEB – додатку.

Якщо в результаті виконання модифікованого запиту WEB – додаток обробляє модифікований параметр, то в даному випадку можна ідентифікувати вразливість HTTP Parameter Pollution.

Якщо в результаті виконання даного запиту модифікований параметр не обробився і запит відобразив правильну інформацію – даний параметр не має вразливості HTTP Parameter Pollution.

У випадку, коли передається декілька параметрів, потрібно дослідити поведінку WEB – додатку у разі модифікації кожного з параметрів по черзі.

Наприклад:

http://host/index.php?par1=val1&par2=val2

Формується ряд модифікованих запитів:

http://host/index.php?par1=val3&par1=val1&par2=val2

http://host/index.php?par2=val4&par1=val1&par2=val2

Ідентифікація наявності вразливості виконується аналогічно із ідентифікацією вразливості для одиничного параметра у запиті.

Важливо пам'ятати, що кожен факт наявності вразливості HPP повинен бути оформлений як дефект захищеності WEB – додатку.

2.6.4 Тестування захищеності WEB – додатку mutillidae на вразливість HTTP Parameter Pollution

За допомогою даного WEB – додатку користувач може обрати будь – який мережевий сканер, який він використовує найчастіше (рисунок 2.11).

1 Згідно наданих специфікацій даний WEB – додаток написаний із використанням технології PHP та застосовує WEB – сервер Apache 2.4.2 або використовується сканування за допомогою Zenmap.

2 На основі даних, наведених у таблиці 1 робиться висновок, що для даної технології та типу серверу у HTTP – запитах фільтрується тільки останній параметр, а перший параметр відкидається.

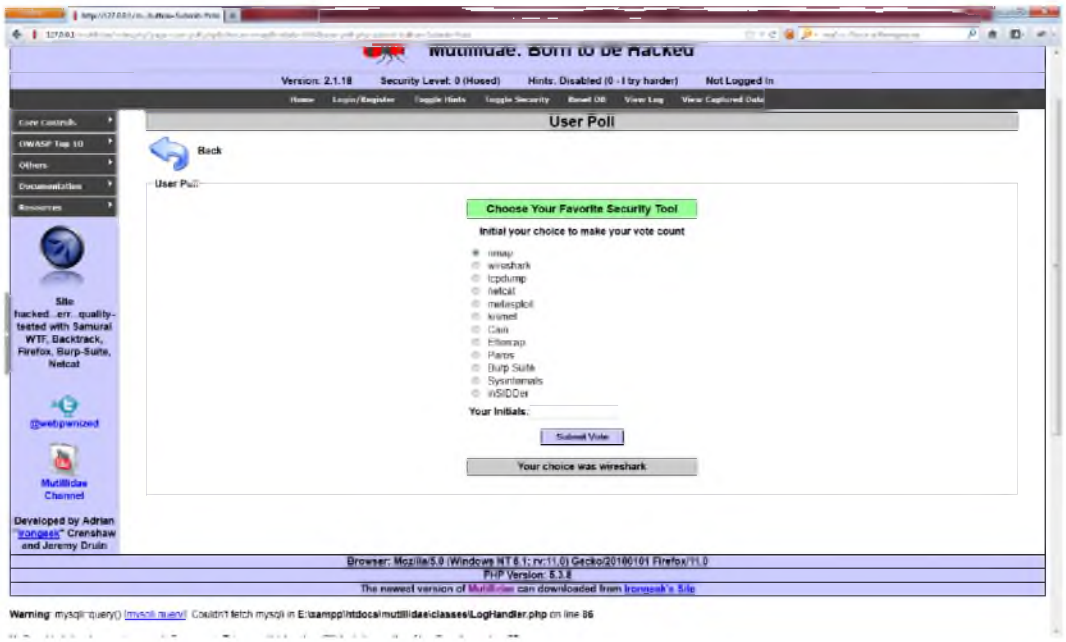


Рисунок 2.11 – Загальний вигляд додатку для тестування

Досліджується «нормальна» робота WEB – додатку. Для цього використається проксі - сервер Paros Proxy. (або WebScarab). На рисунку 2.12 наведений перелік діючих запитів у додатку, отриманий за допомогою програмного продукту WebScarab:

Method	Host	Path	Parameters	Status
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=inSIDDer&initials=&user-poll-php-submit-button=Submit+Vote	200 OK
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=Sysinternals&initials=&user-poll-php-submit-button=Submit+Vote	200 OK
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=Burp+Suite&initials=&user-poll-php-submit-button=Submit+Vote	200 OK
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=Paros&initials=&user-poll-php-submit-button=Submit+Vote	200 OK
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=kismet&initials=&user-poll-php-submit-button=Submit+Vote	200 OK
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=metasploit&initials=&user-poll-php-submit-button=Submit+Vote	200 OK
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=netcat&initials=&user-poll-php-submit-button=Submit+Vote	200 OK
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=tcpdump&initials=&user-poll-php-submit-button=Submit+Vote	200 OK
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=nmap&initials=&user-poll-php-submit-button=Submit+Vote	200 OK
GET	http://127.0.0.1:80	/mutillidae/index.php	?page=user-poll.php&choice=wireshark&initials=&user-poll-php-submit-button=Submit+Vote	200 OK

Рисунок 2.12 – Перелік діючих запитів у WEB – додатку

Схема роботи додатку:

Користувач обирає програмний продукт для тестування захищеності систем, який він найбільш часто використовує, із запропонованого списку, а також вносить дані про себе у поле «Your Initials». Після цього користувач підтверджує свій вибір натисканням кнопки «Submit Vote». В результаті коректної роботи додатку на сторінці відображається повідомлення із текстом «Your choice was …», де замість пропуску висвітлюється назва обраного

программного продукту із списку. В залежності від обраного продукту, результуюче повідомлення буде змінюватися.

Наприклад:

Обирається варіант – nmap та заносяться дані про учасника голосування – Alex.

Формується запит:

http://127.0.0.1/mutillidae/index.php?page=user-poll.php&choice=nmap&initials=Alex&user-poll-php-submit-button=Submit+Vote

Результат виконання запиту відображається на рисунку 2.13:

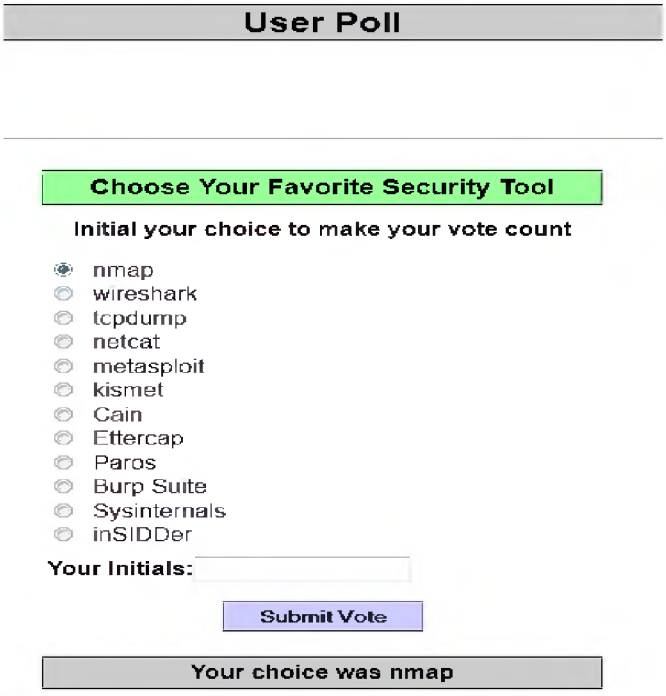


Рисунок 2.13 – Результат голосування у експериментальному додатку

Результуюче повідомлення означає, що був обраний варіант – nmap

3 Виконання тестування на проникнення.

2.7 Формується запит «2» із іншим варіантом для голосування – wireshark.

В результаті отримується запит 2:

http://127.0.0.1/mutillidae/index.php?page=user-poll.php&choice=wireshark&initials=Alex2&user-poll-php-submit-button=Submit+Vote

Згідно логіки роботи додатку користувач повинен бути пере направлений на сторінку, яка буде містити не тільки список для голосування, але й повідомлення, в якому буде записаний обраний варіант. На рисунку 2.14 зображений результат правильного виконання запиту 2:

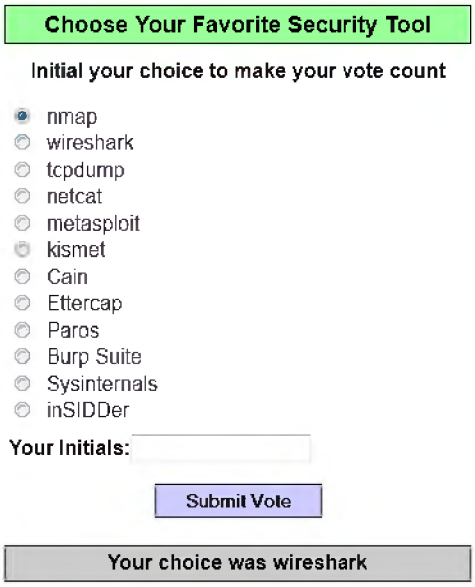


Рисунок 2.14 – Результат правильного виконання запиту 2

За допомогою програмного продукту BurpSuite виконується перехоплення запиту 2, що сформований браузером. На рисунку 2.15 зображений інтерфейс програми BurpSuite у разі перехоплення запиту 2:

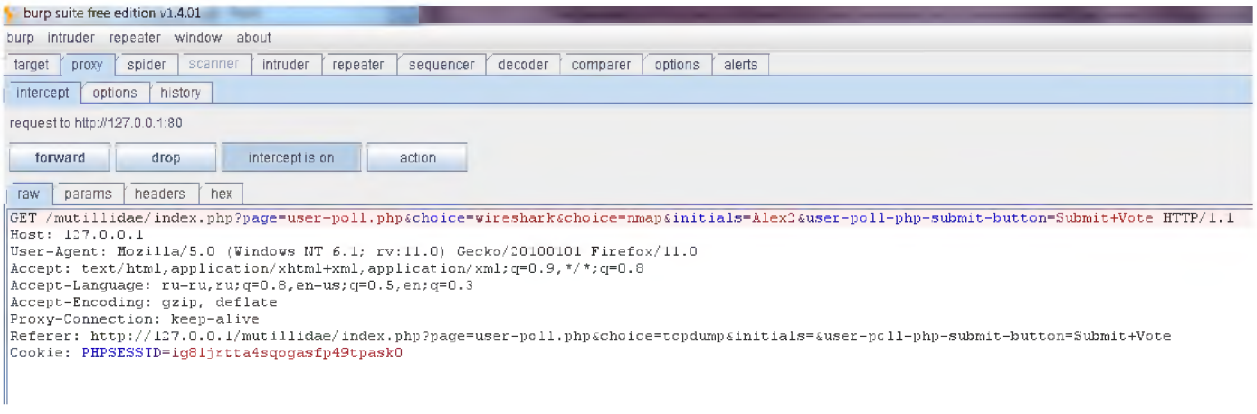


Рисунок 2.15 – Перехоплення запиту за допомогою програмного продукту BurpSuite

2.8 Проводиться модифікація перехопленого запиту. Згідно даних, отриманих на попередніх кроках даної методики, додаток використовує сервер Apache та технологію PHP. Згідно таблиці пріоритетів параметрів

визначається, що в даному експериментальному випадку буде фільтруватися останній параметр, а перший буде відкидатися.

Саме тому модифікація запиту буде включати в себе вставлення у строку запиту параметру choice із значенням - nmap.

Модифікований запит:

http://127.0.0.1/mutillidae/index.php?page=user-poll.php&choice=wireshark&choice=nmap&initials=Alex2&user-poll-php-submit-button=Submit+Vote

Результат виконання модифікованого запиту зображений на рисунку 2.16:

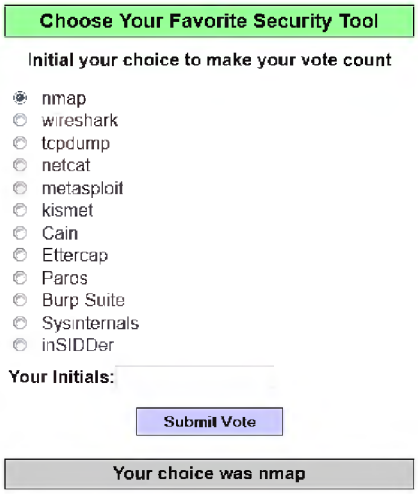


Рисунок 2.16 – Результат виконання модифікованого запиту «2»

Результат виконання модифікованого запиту свідчить про наявність вразливості HTTP Parameter Pollution для параметру із іменем «choice», а також для WEB – додатку в цілому.

Після проведення всіх етапів тестування захищеності кожен знайдений вразливий параметр зазначається у звіт про проведення тестування. Незалежно від форми подання, звіт повинен містити основну інформацію про дефект безпеки:

1 Параметри системи, на якій проводилося тестування;

2 Назва та версія WEB-браузеру на якому виконувалося тестування;

3 Версія WEB-додатку, яка піддавалася тестуванню;

4 Кожний параметр, який є вразливим для HTTP Parameter Pollution;

5 Для кожного вразливого параметру вказується повний шлях для репродукції даного дефекту;

6 Дата тестування;

2.6 Висновок

Проблема інформаційної безпеки WEB-додатку повинна вирішуватися на всіх етапах життєвого циклу додатка.

Тестування безпеки повинно проводитися як на етапі створення окремої функціональної одиниці додатку, так і на етапі системного тестування. Всі знайдені вразливості повинні розглядатися розробниками як дефекти у ПЗ – тому повинні бути усунуті якнайшвидше. Критичність дефектів безпеки повинна бути розподілена так само, як і критичність звичайних функціональних дефектів.

Комплексний підхід до тестування захищеності WEB-додатків дозволить значно зменшити ймовірність порушення цілісності, доступності та конфіденційності інформації, що циркулює у додатку, а також забезпечити логіку його роботи.

Запропонована методика може бути використана як набір рекомендацій для виявлення вразливостей типу HTTP Parameter Pollution, а також у подальшому може бути реалізована як додатковий модуль для вже існуючого сканера вразливостей WEB-додатків.

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Техніко-економічне обґрунтування доцільності розробки методики тестування.

На сьогодні у світі інформаційних технологій та інформаційних послуг однією з загроз є кіберзлочинність. Положення у світі складається таким чином, що на рік збитки від дій кіберзлодіїв сягають до сотень мільярдів доларів.

В галузі кіберзахисту події складаються таким чином, що злочинці завжди за методами та навичками знаходяться попереду спеціалістів з кібербезпеки, а дійсно невразливих інформаційних систем не існує ні за яких умов. А захищеною від зламу вважається лише система ціна атаки на яку значно перевищує ціну збитків, що може завдати ця атака.

Отже у галузі кібербезпеки змогли знайти спосіб бути, якщо не попереду, то хоча б на рівні з кіберзлочинцями. Проведення тестування системи на проникнення дає змогу методами кіберзлодіїв провести реальну атаку на свою ж інформаційну систему, для виявлення найвигідніших векторів атаки, та розробки плану протидії злочинцям. Для проведення подібного роду робіт необхідно використовувати послуги спеціаліста з відповідними специфічними навичками справжнього хакера. Саме для того, що б майбутні спеціалісти з кібербезпеки мали хоча б розуміння та базові навички з проведення тестування був розроблений програмно-апаратно-методичний комплекс з тестування веб-серверу. Крім того, така методика навчання дозволить студенту законно отримати відповідні навички не порушуючи законів.

3.1.1. Отримані навички після проходження завдань

Застосовуючи цю методику співробітник it відділу компанії зможе:

a) Проводити активну мережеву розвідку .

b) Будувати майбутній вектор атаки.

c) Використовувати наявні вразливості системи для отримання повного доступу на фізичною машиною цільової системи.

3.2 Економічна цінність запропонованих до опанування студентами навичок.

Умовно розглянемо та розрахуємо, яку роботу зможе провести співробітник it відділу компанії вже маючи опановані навички запропоновані у цільовій розробці. Детально опишемо економічний ефект від попереднього знаходження подібної вразливості, та підрахуємо, потенційно, якого об'єму збитку вдалося уникнути за допомогою проведення тестування.

3.2.1 Моделювання ситуації

Молода компанія з надання інформаційних послуг, яка має у своєму розпорядженні один сервер на якому працює веб-сервер офіційного сайту компанії. Уявимо, що одна година роботи сайту приносить компанії 2000$ доларів. Недавно з компанії звільнився за власним бажанням адміністратор, що займався обслуговуванням веб-серверу, та добігає кінця термін за який компанія повинна проводити щорічне тестування на проникнення. Отже, далі розрахуємо витрати на проведення тестування та на усунення знайдених недоліків, у розрізі того, що за моделлю сервер має вразливість, що розглядалась у лабораторних роботах, що входять до апаратно-програмно-методичного комплексу.

3.2.2 Розрахунок капітальних витрат

Відштовхуючись від того, що у цьому випадку розглядається конкретний випадок з конкретною вразливістю, то усі підрахунки будуть проводитись відносно саме цієї вразливості.

3.2.2.1 Трудомісткість проведення тестування:

Основна формула:

$$t = t_{тз} + t_{р} + t_{а} + t_{ва} + t_{па} + t_{рт} \quad \text{, годин} \qquad (3.1)$$

$t_{тз}$ – 16 год - тривалість складання технічного завдання .

$t_{р}$ – 6 год - тривалість проведення мережевої розвідки.

$t_{а}$ – 2 год - тривалість процесу аналізу результатів розвідки.

$t_{ва}$ – 5 год - тривалість розробки вектору атаки.

$t_{па}$ – 3 год - тривалість проведення атаки.

$t_{рт}$ – 4 год - тривалість складання звіту про результати тестування на проникнення.

1. Підставляємо числа у формулу:

$$t = t_{тз} + t_{в} + t_{а} + t_{вз} + t_{озб} + t_{овр} + t_{д} = 36 \text{ год.}$$

2. Розрахунок на проведення тестування:

Основна формула:

$$Крп = Ззп + Змч \qquad (3.2)$$

За Ззп візьмемо середню зарплатню пентестувальника за годину, виходячи із розрахунку, що спеціаліст працює 22 дні на місяць по 8 годин на день, отже за місяць спеціаліст працює 172 години на місяць. Середня зарплатня такого спеціаліста становить приблизно 3000 доларів на місяць. Отже за годину роботи такий спеціаліст отримує 17,4 доларів, що при перерахунку в гривню станом на 24 листопада 2020-го року за курсом НБУ становить 493,46 грн за годину роботи такого спеціаліста. Виходячи з розрахунку часу на проведення робіт Ззп на усю роботу становить 17764,56 грн

Ззп – 17764,56 грн

Формула винагороди:

$$З_{зп} = t \cdot З_{пт} \qquad (3.3)$$

Так як, тестування на проникнення проводиться у нашому випадку кваліфікованим спеціалістом, що має право на надання подібних послуг, та відбувається за допомоги власних робочих потужностей тестувальника, то $С_{мч}$ можна знехтувати, адже вона вже включена в ціну проведення тестування на проникнення.

Отже проведення тесту у цьому випадку складає 17764,56 грн.

3.2.3 Прогнозовані збитки від реалізації знайденої вразливості.

Під час тестування спеціаліст з тестування знайшов вразливість, через яку будь хто може завантажувати файли на веб сервер та виконувати їх, що може призвести до відмови в роботі веб-серверу на 6 годин, у зв'язку з роботою зловмисного файлу, що блокує роботу усіх веб-сервісів. Резервного веб-серверу молода компанія не має. Отже потенційні збитки для компанії за простій веб-сайту на 6 годин можуть сягнути 12000$, перераховуючи у гривні за курсом НБУ на 24 листопада 2020-го року, можливі збитки від реалізації загрози становлять 340320 грн

3.2.4 Розрахунки витрат на усунення вразливості та розслідування інциденту

Розраховувати ціну усунення вразливості будемо за формулою

$$С_{ус} = (t_у \cdot З_{сп}) + (t_р + З_{іб}) \qquad (3.4)$$

Де, $t_у$ – 1 год – час за який можна усунути негативні налаштування .

$t_р$ – 6 год – час розслідування інциденту.

$З_{сп}$– ціна роботи за одну годину спеціаліста з обслуговування веб-серверу.

$З_{іб}$ - ціна роботи за одну годину спеціаліста з інформаційної безпеки

Середня заробітна плата спеціаліста з обслуговування систем в Україні становить 25 тис. грн на місяць, отже одна година роботи такого спеціаліста коштує 145,34 грн на годину. Стосовно середня заробітна плата спеціаліста з інформаційної безпеки становить аналогічну суму, а значить $З_{іб}$ = 145,34. Отже підставляємо числа до формули. Отримаємо 1017,38 грн – ціна усунення вразливості.

3.2.5 Підведення загальних витрат.

Для виведення загальних витрат на тестування та усунення знайденої вразливості треба скласти капітальні витрати на проведення пен-тесту та ціну усунення вразливості. Обчислюємо за формулою:

$$С_{заг} = С_{ус} + Крп \qquad (3.5)$$

Де, $С_{заг}$– загальні витрати .

$С_{ус}$ – 6 год – ціна усунення вразливості.

Крп –капітальні витрати на проведення тестування на проникнення.

Отже загальні витрати складають 18781,94 грн .

3.2.6 Ефект від проведення застосування методики

Отже виходячи із можливих збитків від реалізації вразливості, а саме загроза простою сервера протягом 6 годин, що могло б коштувати компанії близько 340320 грн додаючи репутаційні збитки, недієздатності сайту, та коштів що були витрачені для знайдення та усунення вразливості, а саме 18781,94 грн. Потенційно проведення тестування методики дозволило компанії зберегти 322138,06 грн, та уникнути можливих репутаційних ризиків, пов'язаних із реалізацією вразливості

3.3 Висновок

Виходячи з підрахунків вище, спеціаліст із навичками проведення тестування, може допомогти компанії замовнику знайти вразливості в інформаційній системі раніше ніж цією вразливістю зможе скористатись кіберзлочинець, та завдати компанії не тільки грошових збитків, а й репутаційних збитків, що можуть вплинути на дохід компанії у майбутньому.

Отже, виходячи з даних, що вказані вище. Надання співробітнику it відділу компанії навичок проведення тестування на проникнення високо цінується на ринку та є невід'ємною частиною підтримки високого рівня кібербезпеки у сучасних компаніях. З чого можна зробити висновок, що надання таких вмінь студентам, вищий навчальний заклад може значно посилити їх позиції на міжнародному ринку праці.

ПЕРЕЛІК ПОСИЛАНЬ

1 Глушаков С. В., Бабенко М. И., Тесленко Н. С. Секреты хакера. Атака и защита. Учебный курс. – АСТ Москва, 2008. – 544 с.: ил.

2 Бил Дж., Бейкер Э., Казуэлл Б. Snort 2.1 Обнаружение вторжений. – Бином-Пресс, 2009. – 656 с.: ил.

3 Web Server – WikiTweet (Електронний ресурс) / Спосіб доступу: URL: http://www.wikitweet.net/ShowArticle.aspx/Web%20Server-12. – Загол. з екрана.

4 Модели работы веб серверов (Електронний ресурс) / Спосіб доступу: URL: http://algolist.manual.ru/web/servers.php. – Загол. з екрана.

5 Статистика уязвимостей в 2015 году (Електронний ресурс) / Спосіб доступу: URL: http://ptresearch-ru.blogspot.com/2012/03/2011.html. – Загол. з екрана.

6 OWASP: Cross – site Scripting (XSS) (Електронний ресурс) / Спосіб доступу: URL: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS). – Загол. з екрана.

7 Виды информационных угроз и способы борьбы с ними (Електронний ресурс) / Спосіб доступу: URL: http://www.dokwork.ru/2012/01/blog-post.html. – Загол. з екрана.

8 OWASP: LDAP Injection (Електронний ресурс) / Спосіб доступу: URL: https://www.owasp.org/index.php/LDAP_injection. – Загол. з екрана.

9 OWASP: SQL Injection (Електронний ресурс) / Спосіб доступу: URL: https://www.owasp.org/index.php/SQL_injection. – Загол. з екрана.

10 OWASP: XPath Injection (Електронний ресурс) / Спосіб доступу: URL: https://www.owasp.org/index.php/XPATH_injection. – Загол. з екрана.

11 Гипертекстный протокол HTTP (Електронний ресурс) / Спосіб доступу: URL: http://www.intuit.ru/department/network/pami/7/. – Загол. з екрана.

12 HTTP протокол (Електронний ресурс) / Спосіб доступу: URL: http://www.inattack.ru/article/http-protokol/78.html#.T7X7Deg9XK0. – Загол. з екрана

13 НД ТЗІ 2.5-010-03. Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу.

14 НД ТЗІ 2.5-005-99. Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу.

15 НД ТЗІ 2.5-004-99. Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу.

16 Шумский А.А., Шелупанов А.А. Системный анализ в защите информации. – издательство «Гелиос АРВ», 2005. – 244 с.: ил.

17 OWASP AppSec Europe 2009. HTTP Parameter Pollution, May 2009. (Електронний ресурс) / Спосіб доступу: URL: http://www.owasp.org/ images/b/ba/AppsecEU09 _CarettoniDiPaola_v0.8.pdf. – Загол. з екрана.

18 Automated Discovery of Parameter Pollution Vulnerabilities in Web Applications. (Електронний ресурс) / Спосіб доступу: URL: http://www.iseclab. org/people/embyte/ papers/hpp.pdf .– Загол. з екрана.

19 HTTP Parameter Pollution (Електронний ресурс) / Спосіб доступу: URL: http://raz0r.name/articles/http-parameter-pollution/. – Загол. з екрана

20 How to Detect HTTP Paramter Pollution Attacks (Електронний ресурс) / Спосіб доступу: URL: http://www.acunetix.com/blog/whitepaper-http-parameter-pollution/. – Загол. з екрана

21 HTTP Parameter Contamination (Електронний ресурс) / Спосіб доступу: URL: http://www.securitylab.ru/analytics/406673.php – Загол. з екрана

22 HTTP Parameter Pollution (Електронний ресурс) / Спосіб доступу: URL: http://tacticalwebappsec.blogspot.com/2009/05/http-parameter-pollution.html– Загол. з екрана

23 Шереметьєва І.В., Пілова Д.П., Романюк Н.М. Методичні вказівки до виконання економічної частини дипломного проекту (для студентів напряму підготовки 1701 Інформаційна безпека)/ – Дніпропетровськ: ДВНЗ "Національний гірничий університет", 2011. – 17 с.

24 Рейтинг зарплат 2011 року (Електронний ресурс) / Спосіб доступу: URL: http://rada.kosiv.info/kosivs/tidings/8742-reiting-zarplat-2011-roku.html. – Заголовок з екрана.

25 Закон України «Про державну підтримку малого підприємництва» // 2008. – С.1-2.

26 Мєшков В.І. Загальні вимоги до оформлення магістерських дипломних робіт і дипломних проектів спеціалістів для студентів галузей знань 1701 «Інформаційна безпека» та 0509 «Радіотехніка, радіоелектронні апарати та зв'язок». Методичні вказівки. – Д.:Державний ВНЗ «Національний гірничий університет», 2014. – 41 с.

ВИСНОВКИ

Актуальність захисту WEB-додатків від атак, що базуються на недосконалості протоколу передачі даних HTTP стає з кожним роком все більшою. Все більше галузей бізнесу напряму залежать від якості та захищеності своїх WEB-додатків.

Найбільш проблематичною сферою захисту інформаційних ресурсів від несанкціонованого доступу є відсутність стандартизованих підходів щодо безпеки WEB-додатків.

Аналіз реалізації вразливості HTTP Parameter Pollution на стороні клієнта, а також розробка методики підтвердження наявності у WEB-додатках вразливостей цього типу, дасть змогу своєчасно виправляти всі «слабкі» місця у системі безпеки додатку. Чим раніше буде виявлена та виправлена вразливість – тим менші збитки зазнає власник WEB-додатку при реалізації атак.

Розроблена методика тестування повинна бути інтегрована у комплексний процес тестування. В цьому випадку стане можливим більш точна та повна перевірка захищеності окремого додатка. Тестування захищеності в цілому повинно проводитися не тільки на етапі системного тестування додатку, але й вже на етапі тестування окремих його частин.

Розроблена методика тестування розглядає WEB-додаток с позиції користувача. Саме тому головними векторами для перевірки та аналізу у даній методиці стали HTTP-запити від клієнта до WEB-сервера. Даний підхід дозволяє найбільш ефективним способом оцінити можливі ризик компрометації даних та порушення цілісності, доступності та конфіденційності інформації, що циркулює.

Вихідними даними тестування стали вразливі параметри, що передаються у HTTP-запитах. Локалізація та визначення найбільш критичних параметрів дозволить на етапі проектування системи унеможливити використання таких параметрів або переглянути логіку роботи додатку для забезпечення більш надійної передачі параметрів від клієнта до серверної частини додатку.

Економічна ефективність та доцільність впровадження даної методики доводить методика може бути впроваджена, так як збитки від реалізації атаки набагато перевищують вартість методики.

Попередження загроз та атак зі сторони клієнта, що базуються на вразливості HTTP Parameter Pollution основною перевагою якого є можливість обходу WAF (Web Application Firewall), згідно розробленої методики дозволяє забезпечувати захист циркулюючої інформації та розробляти рекомендації для захисту WEB-додатків.

ДОДАТОК А. ПЕРЕЛІК МАТЕРІАЛІВ ДИПЛОМНОЇ РОБОТИ

1 ТИТУЛЬНА СТОРІНКА.
2 РЕФЕРАТ.
3 СПИСОК СКОРОЧЕНЬ.
4 ЗМІСТ.
5 ВСТУП.
6 СТАН ПИТАННЯ.
7 СПЕЦІАЛЬНАЯ ЧАСТИНА.
8 ЕКОНОМІЧНА ЧАСТИНА.
9 ВИСНОВКИ.
10 ПЕРЕЛІК ПОСИЛАНЬ.
11 Додаток А.
12 Додаток Б.
13 Додаток В.
14 Додаток Г.
15 Презентація дипломної роботи.
16 Оптичний носій.

ДОДАТОК Б. ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В. ВІДГУК НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

на тему: «Вибір методики тестування захищеності web – додатків від мережевих атак з можливістю обходу міжмережевих екранів що заходяться в середовищі Інтернет» студента групи 125м-20-1 Грицюка Сергія Ігоровича.

Кваліфікаційна робота за спеціальністю 125 «Кібербезпека» Грицюка С.І. представлена пояснювальною запискою на стор., містить рис., табл., додатка, джерела.

Об'єкт досліджень – процес тестування захищеності WEB-додатків за допомогою методів «чорного» ящика.

Предмет досліджень –методика тестування захищеності WEB-додатків на вразливість HTTP Parameter Pollution..

Мета – обґрунтувати вибір методики тестування захищеності web – додатків від мережевих атак з можливістю обходу Web Application Firewall та практично довести ефективність запропонованої методики.

Наукова новизна результатів, що очікуються, полягає у вирішенні проблеми тестування захищеності WEB-додатків від атак зі сторони клієнта на основі використання методики, що включає в себе використання програмних засобів та виконання тестування на проникнення.

В економічному розділі виконаний розрахунок економічної ефективності запропонованих рішень.

В якості недоліків слід відзначити наступне: недотримання графіка проведення розробки, нечіткість окремих висновків і визначень.

В цілому кваліфікаційна робота виконано у відповідності до вимог, які пред'являються до кваліфікаційних робіт магістра і заслуговує оцінки "добре", а Грицюк Сергій Ігорович присвоєння йому кваліфікації магістра із організації інформаційної безпеки.

Керівник роботи к.т.н., доц.,Флоров.С.В