

Relazione di Tirocinio

Politecnico di Torino - Security Reply Srl

13 Marzo / 26 Aprile 2019



Valerio Casalino

233808

Questa relazione è disponibile in formato digitale su: <http://bit.ly/vcasalino-tirocinio-2019>

Basato sul template (cc|by) di Jan Koornstra 2014-2019 - Xuan Luo, Merel de Jong

Indice

1	Definizione degli obiettivi	1
1.1	Pre-requisiti	1
1.2	Conoscenze da ottenere	1
1.3	Traguardi particolari (opzionali)	2
1.4	L'azienda	2
I	Studio di teoria	3
2	Slow start	4
2.1	Materiale	4
2.2	Vuln Assessment & Pen Test	4
2.3	Information Gathering	5
2.4	Set up dell'ambiente	6
3	WebGoat e OWASP Top Ten	7
3.1	OWASP Top 10: SQL Injection	7
3.1.1	SQL Injection: elementare	7
3.1.2	SQL Injection: Intermedio (e metodi di mitigazione)	8
3.2	OWASP Top 10: XXE (XML eXternal Entity) Attack	9
3.2.1	Cos'è una XML Entity?	9
3.2.2	XXE DoS Attack	10
3.3	OWASP Top 10: Secutity Misconfiguration	11
3.3.1	Authentication Bypasses	11
3.3.2	JWT	11
3.4	OWASP Top 10: XSS (Cross Site Scripting)	11
3.4.1	Reflected XSS	11
3.4.2	DOM-Based XSS	12
3.4.3	Stored (o Persistent) XSS	12
3.5	WebGoat: Access Control Flow	12
3.5.1	Insecure Direct Object Reference	13
3.6	WebGoat: Insecure Communication	13

3.6.1	Proprietà di una comunicazione sicura	13
3.6.2	Crittografia(e)	14
4	Difesa e Contromisure	15
4.1	A1: SQL Injection – How to prevent	15
4.2	A2: Broken Authentication	16
4.3	A3: Sensitive Data Exposure	16
4.4	A4: XML External Entity (XXE)	17
4.5	A5: Broken Access Control	18
4.6	A6: Security Misconfiguration	18
4.7	A7: Cross-Site Scripting (XSS)	19
4.8	A8: Insecure Deserialization	20
4.9	A9: Using Components with Known Vulnerabilities	20
4.10	A10: Insufficient Logging & Monitoring	21
II	Allenamento	22
5	Piattaforma di pratica: Hackthebox	23
5.1	Descrizione	23
5.2	Accesso con Invite Code	24
5.2.1	Tentativo 1: Injection	24
5.2.2	Tentativo 2: Token analysis	24
5.2.3	Tentativo 3: Code analysis	25
5.3	Challenge #1: Netmon	27
5.3.1	User Access	27
5.3.2	Root Access	27
6	Piattaforma di pratica: Root-me	29
6.1	Descrizione	29
6.2	Challenge #1: FTP Authentication	29
6.3	Challenge #2: TELNET Authentication	30
6.4	Challenge #3: ETHERNET Frame	30
III	L'azienda e l'attività	31
7	Webinar:Frida Framework	32
7.1	Introduzione	32
7.2	What is FRIDA?	32
7.3	What is it for?	32

7.4	Installation and Basic Usage	32
7.5	Frida Cookbook	33
8	Reportistica	34
8.1	Common Vulnerability Scoring System	34
8.2	CVSS2 Score	34
8.3	Metrica di base	34
8.4	Access Vector (AV)	35
8.5	Access Complexity (AC)	35
8.6	Authentication (Au)	36
8.7	Confidential Impact (C)	37
8.8	Integrity Impact (I)	38
8.9	Availability Impact (A)	38
8.10	Livelli di Severità	38
8.11	Esempio	38
8.12	Riferimenti	39
9	Attività Reply	41
9.1	Introduzione	41
9.2	Organizzazione e struttura	41
9.3	Nuove competenze acquisite	41
9.4	Conclusione	42
	Bibliography	44

Capitolo 1

Definizione degli obiettivi

1.1 Pre-requisiti

I prerequisiti per capire e progredire velocemente nell'apprendimento e nelle dinamiche sono:

- Conoscenza di architetture di reti (trattato all'esame di Reti di Calcolatori).
- Conoscenza basilare del funzionamento delle query SQL (trattato all'esame di Basi si Dati).
- Conoscenza molto basilare della logica di programmazione e algoritmica.

1.2 Conoscenze da ottenere

Ciò che è necessario apprendere all'inizio di questa esperienza è:

- Architettura di una web application.
- Conoscenza più approfondita delle formulazioni delle query SQL.
- Funzionamento di XML.
- Javascript.
- JSON.

Questi argomenti, seppure non trattati estensivamente, sono necessari per comprendere il lavoro di un penetration tester, o, generalmente di chi lavora nel campo della cyber security.

1.3 Traguardi particolari (opzionali)

Entro la fine di questo tirocinio dovrei essere in grado di eseguire un penetration test / vulnerability assesment su una piattaforma reale (bug bounty), ed eventualmente, con un po' di fortuna, segnalare uno 0day (vulnerabilità scoperta la prima volta) ottenendo un riconoscimento.

1.4 L'azienda

Reply S.p.A. è una società italiana di consulenza, system integration, applicazioni di digital services, specializzata nella progettazione, implementazione e manutenzione di soluzioni basate su Internet e sulle reti sociali.

La società è quotata presso la Borsa valori di Milano dove è presente negli indici FTSE Italia Mid Cap e FTSE Italia STAR.

L'organizzazione della holding Reply è peculiare, perché per ogni segmento di business viene partorita un'altra sotto-azienda. Nel caso specifico, per quello che riguarda la consulenza nel campo security, l'azienda è Security Reply Srl, che è quella per cui sto svolgendo il tirocinio.

Lo spazio interno nella sede in [via Giorgione 59 Roma \(RM\)](#) è suddiviso in piani, e si accede col tesserino ad ogni piano. All'ultimo c'è la reception e la sezione Security è al primo piano. Io sono stato assegnato ad uno spazio nell'open space insieme ad un'altra decina di persone fisse e altre variabili.

Parte I

Studio di teoria

Capitolo 2

Slow start

2.1 Materiale

Il materiale di riferimento da studiare e le linee guida da seguire per questo tirocinio:

- Un documento chiamato "OWASP top 10" [10], il quale definisce le 10 principali vulnerabilità delle web applications in ordine combinato di gravità e frequenza. Il documento è disponibile con licenza Creative Commons.
- Un libro chiamato "Hacklog Volume 2" [8], il quale sarà come il mio testo di riferimento per l'attività del tirocinio. Disponibile in licenza Creative Commons.
- Una lista di link a piattaforme online di test e training su temi di WAPT e analisi infra-strutturale.

2.2 Vuln Assessment & Pen Test

Le Web Applications sono costituite da una parte di back-end, front-end e database, ognuna di queste possiede delle criticità specifiche, le OWASP top 10 sono le vulnerabilità più frequenti dei sistemi connessi in rete. Il compito di chi si occupa della sicurezza di questi sistemi può essere quello di generare un **Vulnerability Assessment**, come quello di effettuare un vero e proprio **Penetration Test**. La differenza tra un Vulnerability Assessment e un Penetration Test è che il primo si effettua per ottenere una lista di vulnerabilità trovate analizzando la web application, il secondo è un test vero e proprio della sicurezza della stessa.

- **Black box**: la modalità per cui il tester non conosce nulla di come sia fatta la web app, ma è anche il caso che più si avvicina alla realtà.

- **White box:** la modalità per cui il tester conosce tutto il codice sorgente della web app.
- **Gray box:** la modalità mista tra le 2 precedenti.

Delle vulnerabilità osservate, mi piacerebbe approfondire, dopo un approfondimento sugli strumenti da utilizzare, la SQL injection, Sensitive data Exposure, *Security Misconfiguration*.

2.3 Information Gathering

Per un Hacker attaccante, l'information gathering è una fase molto importante per scoprire le vulnerabilità di un sistema. Consiste in un'attività di raccolta di informazioni al fine di fare profiling.

Un primo passo molto basilare per iniziare a fare information gathering è quello di effettuare un whois del dominio: serve a scoprire a chi è associato è un certo dominio, chi se ne occupa, chi è l'intestatario, ecc. Infatti, tramite il comando su shell Linux:

```
$ whois reply.com
```

e si otterrà l'output in figura 2.1.

```
Domain:                reply.it
Status:                ok
Signed:                no
Created:                1996-09-18 00:00:00
Last Update:           2018-12-31 00:45:27
Expire Date:           2019-12-15

Registrant
Organization:          Reply S.p.A.
Address:                C.so Francia 110
                        Torino
                        10147
                        TO
                        IT
Created:                2009-12-15 16:14:50
Last Update:           2014-06-11 13:20:11

Admin Contact
Name:                  Angelucci Daniele
Address:                corso Francia 110
                        Torino
                        10143
                        TO
                        IT
Created:                2014-06-11 13:08:06
Last Update:           2014-06-11 13:08:06

Technical Contacts
Name:                  Technical Support
Organization:          Register.it S.p.A.
Address:                Via Zanchi 22
                        Bergamo
                        24126
                        BG
                        IT
Created:                2009-09-28 11:01:09
Last Update:           2012-04-27 15:13:45

Registrar
Organization:          Register.it s.p.a.
Name:                  REGISTER-REG
Web:                   http://we.register.it
DNSSEC:                no

Nameservers
dns2.reply.it
dns1.reply.it
dns4.reply.it
```

Figura 2.1: Esempio di output del comando whois.

Se gli IP originali dovessero essere coperti da servizi di reverse proxy, firewall o altro, ci sono diversi metodi non trattati nello specifico su come ricavare l'indirizzo IP del server richiesto.

2.4 Set up dell'ambiente

Per mia comodità, ho installato una versione live di Kali Linux su una USB drive [1]. L'obiettivo è simulare una web application con OWASP WebGoat [11], che è “una web application deliberatamente insicura, mantenuta da OWASP e progettata per fornire lezioni sulla sicurezza delle web applications”.

Per imparare tramite la piattaforma è necessario avvalersi di altri strumenti di intercettazione del traffico in stile man-in-the-middle [17]. Per fare ciò, tutte le distribuzioni proposte per il penetration testing offrono, tra la pletora degli strumenti forniti per lo scopo, sicuramente almeno uno tra i seguenti:

- **Burp Suite** [download link]: il più utilizzato, ma dal codice proprietario.
- **OWASP Zap** [GitHub link]: meno utilizzato, ma disponibile open source.

Io ho adottato Burp come strumento per imparare. Per usare efficacemente gli strumenti, ho messo WebGoat in ascolto sulla porta 8080, ma il traffico uscente l'ho direzionato sulla porta 8000, sulla quale è in ascolto Burp. Tutto ciò è complesso perché avviene in locale. In un caso reale non ci sarebbe bisogno di re direzionare il traffico in questo modo, ma è il caso di effettuare questi test offline, perché durante questa fase di apprendimento la macchina diventa estremamente vulnerabile ad attacchi esterni.

Capitolo 3

WebGoat e OWASP Top Ten

3.1 OWASP Top 10: SQL Injection

3.1.1 SQL Injection: elementare

Per iniziare, le SQL Injection basilari sono quelle che si possono realizzare quando le query SQL in un applicativo web sono scritte nella forma:

```
"select * from users where LAST_NAME='  
                                " + userNameInput() + " ' "  
"select * from users where USER_ID=" + userIdInput()
```

Dove `userNameInput()` (come `userIdInput()`), solitamente è una funzione che legge l'input da un box html dove si inseriscono i dati. La query risultante viene inviata al Database, il quale poi risponderà senza ulteriori controlli.

L'injection in questi casi è ciò che si mette nel box per far fare al DB ciò che si vuole. Nell'esempio specifico in figura 3.1 otterremo la lista di tutti gli user senza avere alcuna informazione.



Figura 3.1: Schermata di successo del primo esercizio di SQL Injection.

Con la stessa logica si esegue l'Injection numerica (senza gli apici ' che delimitano la stringa).

3.1.2 SQL Injection: Intermedio (e metodi di mitigazione)

I metodi migliori per evitare le vulnerabilità del paragrafo precedente sono i seguenti:

- **Immutable queries:** sono quelle che non variano mai e non possono essere interpretate.
- **Static queries:** sono quelle che variano solo per la sessione in corso:

```
select * from products;
select * from users where user = "'" +
    session.getAttribute("UserID") + "'";
```

- **Parameterized queries:** come dice il nome, sono quelle parametrizzate:

```
String query = "SELECT * FROM users
                WHERE last_name = ?";
PreparedStatement statement =
    connection.prepareStatement(query);
statement.setString(1, accountName);
ResultSet results = statement.executeQuery();
```

- **Stored procedures:** sono quelle che devono passare per una serie di procedure definite dalla web application, sono “safe” solo se evitano di generare SQL dinamico.
- **Limitazione dei privilegi:** aggiungere un layer di controllo in modo che alcune query possano essere eseguite solo da utenti/amministratori con privilegi adeguati.

Buona norma è comunque validare l’input perché previene anche altri tipi di vulnerabilità che vedrò più avanti. Ma il fatto che vengano prese queste precauzioni potrebbe non essere comunque abbastanza. Basta una query organizzata con un “order by” per inserire una select in-terna alla condizione di ordinamento, in tal modo, sempre tramite una SQL Injection, è possibile ricavare delle informazioni non necessariamente messe a disposizione del client, ma presenti nel DB.

Per imparare questo concetto, mentre ero seguito, ho svolto l’esercizio otto della sezione SQL Injection (mitigation) di WebGoat. Questo prevedeva una tabella di server e mi chiedeva di ricavarne l’ip di un server non in lista.

Analizzando il traffico e le richieste HTTP, si poteva chiaramente vedere che, quando venivano ordinati i server nella tabella, appariva, nell'header un parametro tipo `...servers?column="hostname"`. Quindi la query sarà stata qualcosa del tipo:

```
select * from servers ... order by "hostname"
```

Ma in questo modo è possibile inserire, al posto di `hostname`, una porzione SQL del tipo:

```
(case when exists
(
select id from servers
where hostname='nome_che_vuoi' and
substring(ip,start_index,end_index)=
      'ip_da_indovinare'
)
then id else ip end)
```

In questo modo non stiamo esplicitamente richiedendo l'ip del server incognito, ma stiamo ordinando quelli che già abbiamo a seconda della correttezza dell'ip che stiamo cercando di indovinare volta per volta. Se stiamo indovinando la porzione dell'ip, allora i server presenti saranno ordinati in ordine di id, altrimenti saranno ordinati in ordine di ip. (la scelta del modo di ordinare è arbitraria).

3.2 OWASP Top 10: XXE (XML eXternal Entity) Attack

3.2.1 Cos'è una XML Entity?

Un'entità XML può essere definita permette di definire dei tags che saranno poi analizzati in fase di esecuzione. Questi tags possono riferirsi ad entità interne, esterne o parametrizzate.

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE author [
    <!ELEMENT author (#PCDATA)>
    <!ENTITY js "Jo Smith">
]>
<author>&js;</author>
```

Nella porzione di codice, l'analizzatore sostituirà "&js;" con l'entità "Jo Smith". Una XXE injection funziona definendo una nuova entità nel pacchetto http che viene inviato al server (tramite un proxy come Burp).

3.2.2 XXE DoS Attack

Con le XXE Injection si possono performare degli attacchi DoS (Denial of Service), il cui più famoso è definito Billion Laughs [14], e il codice da iniettare nell'XML per realizzarlo è il seguente:

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ELEMENT lolz (#PCDATA)>
  <!ENTITY lol1 "&lol;&lol;&lol;
                &lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol2 "&lol1;&lol1;&lol1;
                &lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;
                &lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;
                &lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;
                &lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;
                &lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;
                &lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;
                &lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;
                &lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]> <lolz>&lol9;</lolz>
```

Questo verrà analizzato dall'XML parser ricorsivamente, generando molti dati (circa 3GB).

3.3 OWASP Top 10: Security Misconfiguration

3.3.1 Authentication Bypasses

Una non corretta configurazione di un 2 Factor Authenticator può portare a problemi di sicurezza più severi di quelli che si cerca di prevenire: solo nel 2016 PayPal aveva un problema di questo genere a causa di una security misconfiguration [4]. Nella POST http(s) generata da una pagina di richiesta delle “domande di sicurezza” è stato possibile cancellare la parte venivano fornite le risposte alle 2 domande, dando la possibilità di reimpostare la password “autenticandosi senza autenticarsi”.

3.3.2 JWT

I JWTs (JSON Web Token) sono delle porzioni di dati in formato JSON che contengono informazioni sull'utente in sessione. Sono composti da 3 parti:

1. **Header:** contenente informazioni sulla codifica e altro.
2. **Payload:** contenente le informazioni principali (id utente, permessi etc...).
3. **Firma:** firma digitale del server con cui codificare le informazioni secondo l'algoritmo descritto dall'header.

Questi si presentano nelle richieste HTTP (GET o POST) sotto la voce “cookie”.

Se si configura male la transazione, è possibile intercettare il pacchetto HTTP e decodificando le varie sezioni, si possono modificare i permessi di sessione, ad esempio per acquisire diritti di amministratore sulla richiesta effettuata.

3.4 OWASP Top 10: XSS (Cross Site Scripting)

“Gli attacchi di tipo XSS aprono un mondo completamente nuovo nel vasto universo della Web Security; essi infatti sono tanto pericolosi da soli quanto devastanti in appoggio a vulnerabilità anche più gravi”

Stefano Novelli - Hacklog vol.2

3.4.1 Reflected XSS

Le vulnerabilità di tipo reflected XSS (figura 3.2) sono comuni da ritrovare in diversi ambienti dinamici come pagine d'errore, pagine di ricerca e qualunque altra pagina in cui l'informazione viene manipolata ma non viene memorizzata.

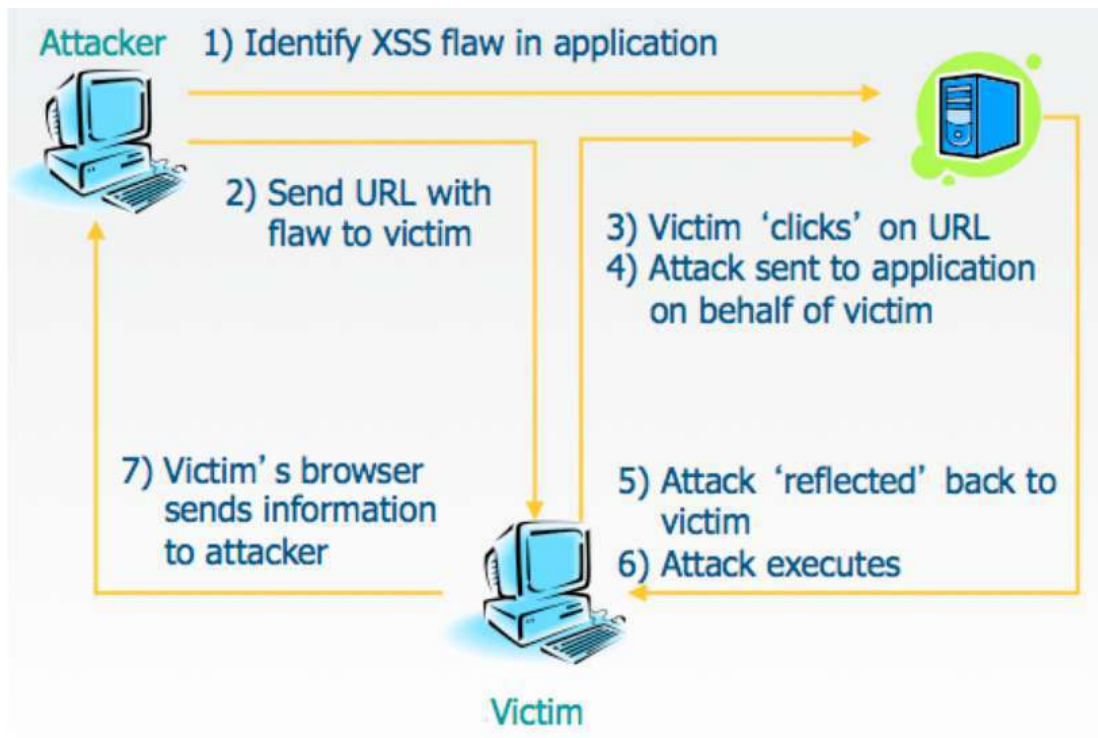


Figura 3.2: Schema rappresentativo del funzionamento delle reflected XSS.

Cliccando su un link che contiene del codice eseguibile è possibile far eseguire il codice al client (secondo lo schema 3.2). Rubando session tokens o qualsiasi informazione si voglia.

3.4.2 DOM-Based XSS

Gli attacchi di tipo DOM based (che sono reflected in fondo) sfruttano gli elementi DOM [16] presenti nel client piuttosto che sul server.

3.4.3 Stored (o Persistent) XSS

Le XSS stored vengono memorizzate all'interno di un database o di un file. Questo solitamente è causato da una vulnerabilità che permette di iniettare codice HTML o Javascript che verrà poi salvato e successivamente recuperato dalla web app (ad esempio la firma in un forum, una chat o oppure un guestbook). Per far cadere nel tranello la vittima basterà reindirizzarla alla pagina violata.

3.5 WebGoat: Access Control Flow

Questa categoria racchiude tutte le vulnerabilità legate all'accesso utente a causa di una configurazione approssimativa dell'accesso alle informazioni su una web application.

3.5.1 Insecure Direct Object Reference

Una Direct Object Reference avviene nel momento in cui l'applicazione usa un input fornito dal client per accedere ad oggetti e dati. Alcuni esempi dove basta una richiesta GET sono i seguenti:

```
https://some.company.tld/dor?id=12345  
https://some.company.tld/images?img=12345  
https://some.company.tld/dor/12345
```

Ma è possibile puntare a questi oggetti anche con richieste di altro tipo. Questo genere di reference diventa insicura nel momento in cui l'oggetto puntato è un profilo, o un dato sensibile. Questo è molto probabile quando si segue un “**RESTful pattern**” [9] per la manipolazione delle risorse di una web app.

3.6 WebGoat: Insecure Communication

Quasi inutile dirlo, la comunicazione crittografata è fondamentale per la confidenzialità dei dati (come per il login su una piattaforma qualsiasi).

3.6.1 Proprietà di una comunicazione sicura

Dato come esempio un mittente ed un destinatario, perché la comunicazione tra i due risulti sicura devono essere soddisfatti i seguenti requisiti:

- **Confidenzialità:** solo il mittente e il destinatario devono essere in grado di capire il contenuto del messaggio trasmesso (Crittografia).
- **Integrità del messaggio:** vogliamo che il messaggio non sia alterato, che sia questo di proposito da un malintenzionato o a causa di un errore.
- **Autenticazione degli interlocutori:** sia il mittente che il destinatario devono potersi identificare a vicenda.
- **Sicurezza operativa:** in sostanza quelle misure di sicurezza che si occupano di monitorare il traffico che passa dalla rete pubblica alla rete privata (Firewall).

Questi requisiti sono applicati anche alla sicurezza della rete. In una connessione TCP vengono scambiati sia segmenti dati che segmenti di controllo (normalmente cifrati).

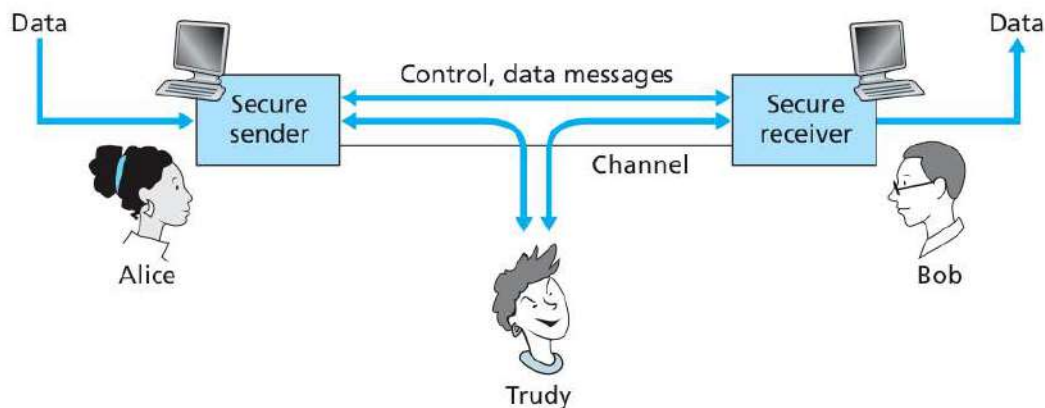


Figura 3.3: (not so) secure communication.

Se un intruso (Trudy nell'immagine 3.3) potesse eludere i requisiti per la comunicazione sicura potrebbe:

- Fare sniffing e registrare i segmenti di dati e di controllo.
- Modificare, inserire o cancellare alcuni messaggi o il loro contenuto.

Grazie a queste facoltà un intruso potrebbe performare una grande quantità di attacchi mirati: spiare conversazioni, rubare identità, impedire l'accesso ad un servizio etc... Avendo il quadro della situazione, vediamo che ruolo gioca la crittografia e perché sia diventata così centrale nella sicurezza della rete.

3.6.2 Crittografia(e)

Nei sistemi moderni vengono utilizzati algoritmi standardizzati e conosciuti anche dal potenziale intruso. Chiaramente se tutti quanti conoscono l'algoritmo deve esistere qualcosa che tenga le informazioni segrete: questa cosa è la chiave. Nella crittografia a chiave simmetrica il mittente avrà la chiave di cifratura e il destinatario avrà quella di decifratura. Nella crittografia a chiave pubblica viene usata una coppia di chiavi, una pubblica e l'altra o al mittente o al destinatario.

Esempi di crittografia a chiave privata sono:

- Cifrario di Cesare.
- Cifrario mono-alfabetico / polialfabetico.
- Cifrario a blocchi.
- Cifrario a catena di blocchi.

L'algoritmo di cifratura a crittografia asimmetrica per antonomasia è l'RSA.

Capitolo 4

Difesa e Contromisure

In questa sezione prendo in considerazione le remediation trovate in rete e/o riportate sul mio libro di testo (Hacklog volume 2) e/o sul documento “OWASP Top Ten” del 2017, in quanto l’azienda non si occupa nello specifico dei fix, ma può capitare e comunque va avuta una conoscenza generale dell’argomento.

4.1 A1: SQL Injection – How to prevent

La soluzione proposta dall’OWASP è quella di mantenere i dati di input e il codice che esegue le query separati, per farlo le soluzioni sono diverse:

- Affidarsi a delle API che parametrizzano l’input, oppure a degli Object Relational Mapping Tools [18], i quali associano degli oggetti astratti a delle strutture dati implementate in diversi linguaggi di programmazione o con diverse convenzioni.
- Usare delle “whitelist” per validare l’input lato server, anche se non è una soluzione sicura perché alcune API richiedono l’utilizzo di caratteri speciali nell’input.

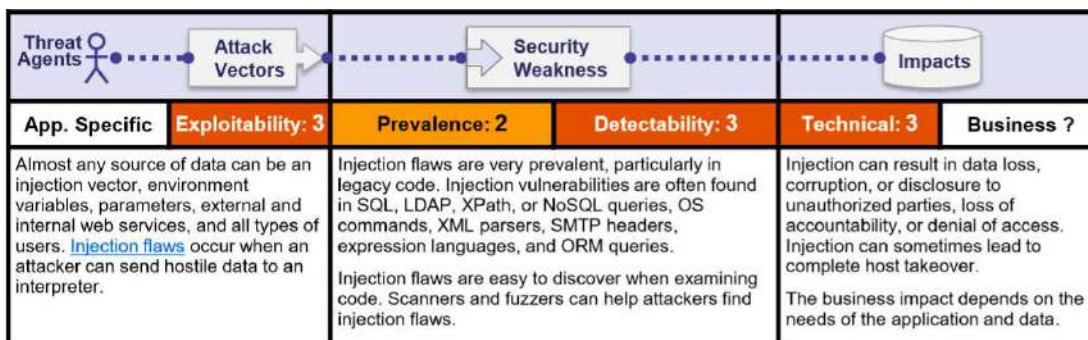


Figura 4.1: Synthesis of SQL Injection characteristics

4.2 A2: Broken Authentication

Per il documento dell'OWASP le possibili soluzioni sono:

- Dove possibile, implementare un'autenticazione multi-fattore per prevenire alcuni tipi di attacco, tra cui il brute force.
- Non utilizzare, nel prodotto finale, username e passwords di default, come admin:admin.
- Implementare nel servizio dei controllori per le password deboli.
- Limitare il numero di tentativi per il login in un certo arco di tempo.
- Utilizzare un sistema di session ID e cookie da non passare tramite URL e da invalidare dopo il logout di chi lo usa.

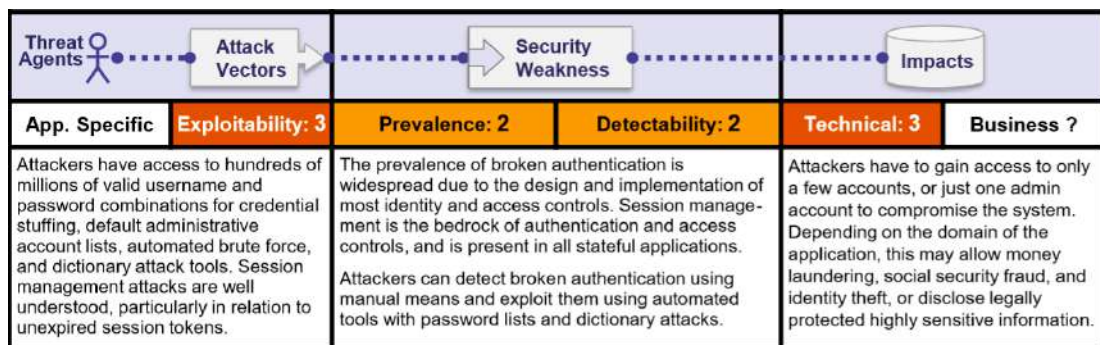


Figura 4.2: Synthesis of Broken Authentication characteristics

4.3 A3: Sensitive Data Exposure

Per il documento dell'OWASP, i requisiti minimi per evitare di esporre dati sensibili degli utenti, e per compiere i primi passi per allinearsi alla GDPR europea, sono:

- Classificare i dati elaborati, salvati o trasmessi dall'applicazione e applicare dei controlli su queste classificazioni.
- Non mantenere più informazioni del necessario per evitare che questi siano esposti.
- Nel transitare sulla rete, i dati devono essere cifrati e gli scambi devono essere effettuati tramite protocolli TLS.
- Impedire che questi dati siano memorizzati in cache.
- Salvare le password cifrandole con un fattore di ritardo, come:

- Argon2.
- Bcrypt.
- Scrypt.
- PBKDF2.





 Threat Agents		 Attack Vectors		 Security Weakness		 Impacts					
App. Specific		Exploitability: 2		Prevalence: 3		Detectability: 2		Technical: 3		Business ?	
Rather than directly attacking crypto, attackers steal keys, execute man-in-the-middle attacks, or steal clear text data off the server, while in transit, or from the user's client, e.g. browser. A manual attack is generally required. Previously retrieved password databases could be brute forced by Graphics Processing Units (GPUs).				Over the last few years, this has been the most common impactful attack. The most common flaw is simply not encrypting sensitive data. When crypto is employed, weak key generation and management, and weak algorithm, protocol and cipher usage is common, particularly for weak password hashing storage techniques. For data in transit, server side weaknesses are mainly easy to detect, but hard for data at rest.				Failure frequently compromises all data that should have been protected. Typically, this information includes sensitive personal information (PII) data such as health records, credentials, personal data, and credit cards, which often require protection as defined by laws or regulations such as the EU GDPR or local privacy laws.			

Figura 4.3: Synthesis of Sensitive Data Exposure characteristics

4.4 A4: XML External Entity (XXE)

L'allenamento degli sviluppatori è indispensabile per identificare e mitigare queste vulnerabilità, ma come regole generali, per prevenire le XXE è necessario:

- Dove possibile, è meglio utilizzare dati in formato meno complesso possibile ed evitare che dati sensibili siano serializzati.
- Aggiornare e aggiustare tutti gli analizzatori di XML che sono in uso nell'applicazione e sul sistema operativo sottostante.
- Non far accettare dall'XML Parser le XML External Entities seguendo il Cheatsheet dell'OWASP [\[GitHub link\]](#).
- Degli strumenti di SAST [\[3\]](#) (Static Application Security Test) possono aiutare a rilevare queste e altre vulnerabilità. Lo strumento di SAST che ho avuto occasione di usare è HP Fortify [\[Website link\]](#).


 <pre>graph LR; A[Threat Agents] -.-> B[Attack Vectors]; B -.-> C[Security Weakness]; C -.-> D[Impacts]</pre>					
App. Specific	Exploitability: 2	Prevalence: 2	Detectability: 3	Technical: 3	Business ?
Attackers can exploit vulnerable XML processors if they can upload XML or include hostile content in an XML document, exploiting vulnerable code, dependencies or integrations.		By default, many older XML processors allow specification of an external entity, a URI that is dereferenced and evaluated during XML processing. SAST tools can discover this issue by inspecting dependencies and configuration. DAST tools require additional manual steps to detect and exploit this issue. Manual testers need to be trained in how to test for XXE, as it not commonly tested as of 2017.		These flaws can be used to extract data, execute a remote request from the server, scan internal systems, perform a denial-of-service attack, as well as execute other attacks. The business impact depends on the protection needs of all affected application and data.	

Figura 4.4: Synthesis of XXE vulnerabilities characteristics

4.5 A5: Broken Access Control

Il controllo dell'accesso è efficace solo se implementato lato server con codice sicuro e fidato, oppure tramite API che non interagiscono con un server, in modo che non possa essere modificato il modo di accertare l'accesso o i metadati.

Per prevenire questo genere di vulnerabilità:

- Con l'eccezione delle risorse pubbliche, rifiuta l'accesso per default.
- Implementare meccanismi di accesso una volta sola in modo che siano riutilizzati tra le applicazioni.
- Il controllo d'accesso deve essere rinforzato con il controllo di appartenenza all'utente, piuttosto che permettere che chiunque possa creare, leggere, cancellare ogni record.
- Mantenere i log per gli accessi non riusciti e notificare l'amministratore quando appropriato.
- I JWT tokens dovrebbero essere invalidati al logout.

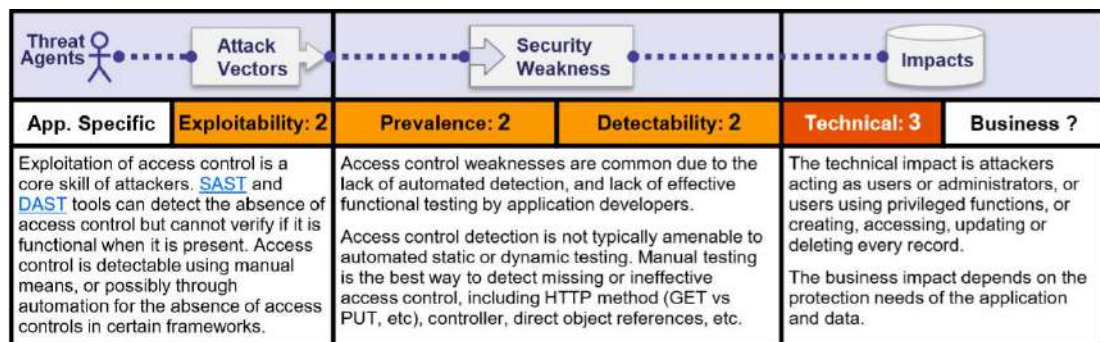


Figura 4.5: Synthesis of Broken Access Control characteristics

4.6 A6: Security Misconfiguration

Questo rimedio va associato all'A9. Infatti, non è necessario solamente utilizzare frameworks sempre aggiornati e tenere traccia di come essi si comportano, ma è necessario anche configurare correttamente questi frameworks.

Il rimedio principale è quello di utilizzare una piattaforma con il numero minimo di funzionalità, per evitare di avere componenti inutilizzati che possano rappresentare un rischio per la sicurezza dell'intera piattaforma.

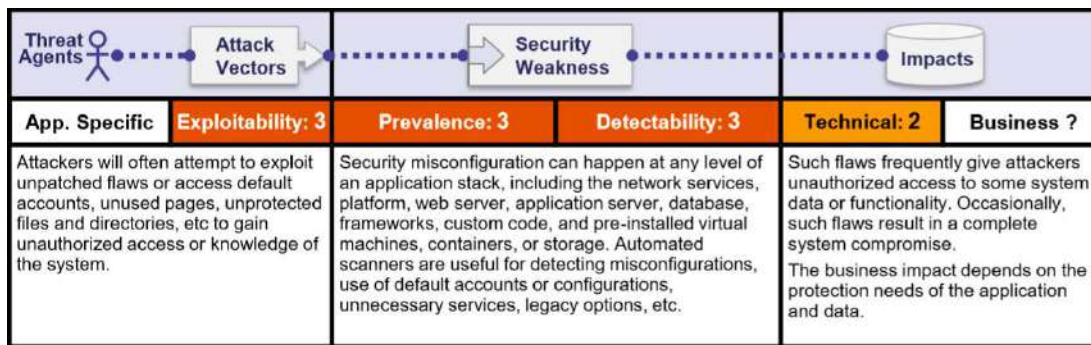


Figura 4.6: Synthesis of Security Misconfiguration characteristics

4.7 A7: Cross-Site Scripting (XSS)

La soluzione proposta dal libro è un filtro per caratteri e tag HTML implementato nell'acquisizione con PHP. Il codice è il seguente:

```
<?php
    // Sanitize message input

    $message = stripslashes($message);
    $message = ...
    $message = htmlspecialchars($message);

    // Sanitize name input
    $name = stripslashes($name);
    $name = htmlspecialchars($name);
?>
```

Presa come esempio l'acquisizione di nome e messaggio da inputbox (per esempio per un form di contatto su una pagina web), la soluzione proposta prevede di usare le funzioni PHP “*stripslashes*” e “*htmlspecialchars*” per rimuovere automaticamente i caratteri che potrebbero appartenere a del codice html o javascript che potrebbe essere eseguito dal browser che accede alla pagina.

Questa soluzione funziona sia per le reflected XSS sia per le stored XSS allo stesso modo. In generale valutare la correttezza dell'input è una buona pratica per lo sviluppo di web applications.

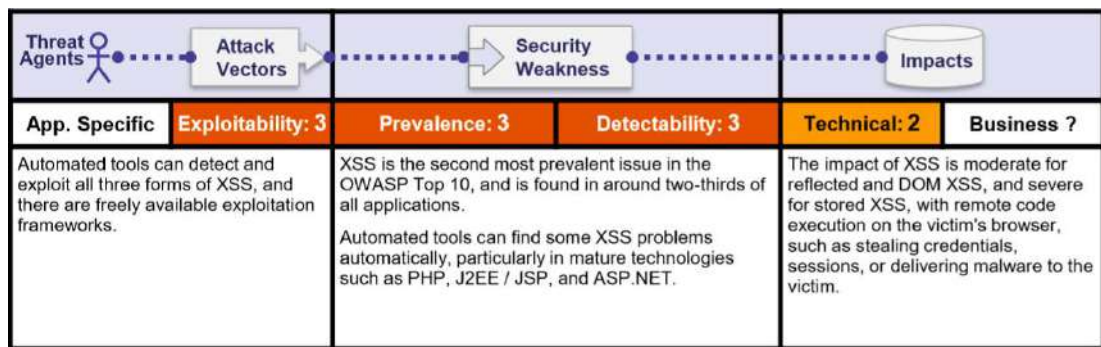


Figura 4.7: Synthesis of XSS characteristics

4.8 A8: Insecure Deserialization

In accordo con il documento dell'OWASP, l'unico modo davvero efficace per evitare questo genere di falle è quello di evitare di accettare oggetti serializzati.

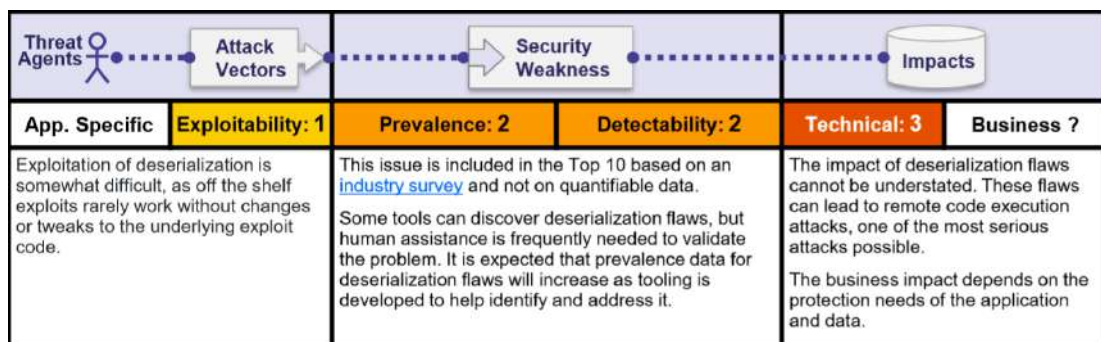


Figura 4.8: Synthesis of Insecure Deserialization characteristics

4.9 A9: Using Components with Known Vulnerabilities

Questo rimedio va associato alla A6. Infatti, non è sufficiente configurare bene il proprio sistema, ma bisogna anche tenere aggiornati tutti i componenti che ne fanno parte, in modo da applicare le patch per le falle di sicurezza scoperte da chi fornisce il framework o il componente.

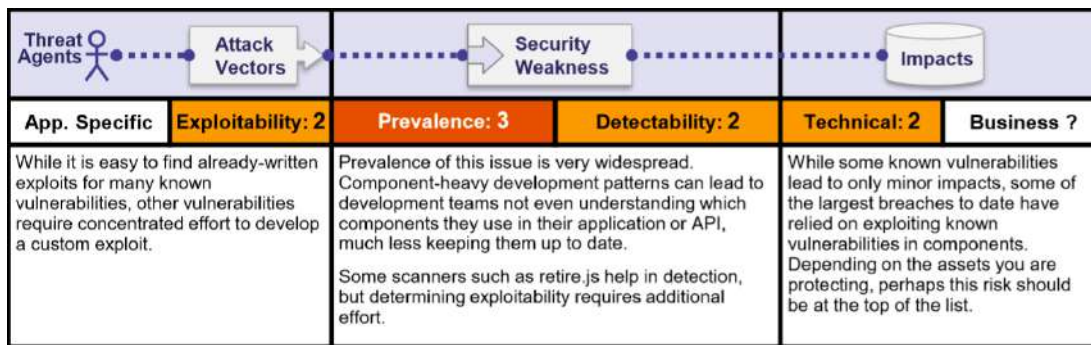


Figura 4.9: Synthesis of Known Vulnerabilities characteristics

4.10 A10: Insufficient Logging & Monitoring

Buona pratica nel gestire un sistema è quella di monitorare a sufficienza gli accessi e gli eventi di chi vi interagisce. Per prevenire la mancanza di log è fare in modo che questi vengano presi.

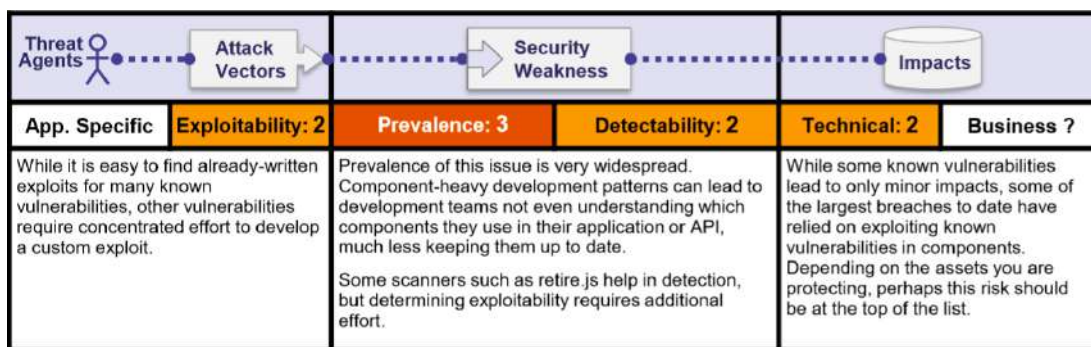


Figura 4.10: Synthesis of Insufficient Logging & Monitoring characteristics

Parte II

Allenamento

Capitolo 5

Piattaforma di pratica: Hackthebox

5.1 Descrizione

La piattaforma <https://www.hackthebox.eu> è una piattaforma di allenamento che permette di accedere ad una rete di server in rete privata (locale) per effettuare analisi infrastrutturali su macchine vulnerabili.

La particolarità di questa piattaforma è che per potersi registrare bisogna avere un invite code, il quale può essere trovato sfruttando qualsiasi conoscenza si abbia. Una volta trovato il codice sarà possibile registrare mail, username e password per il servizio. Iniziamo!

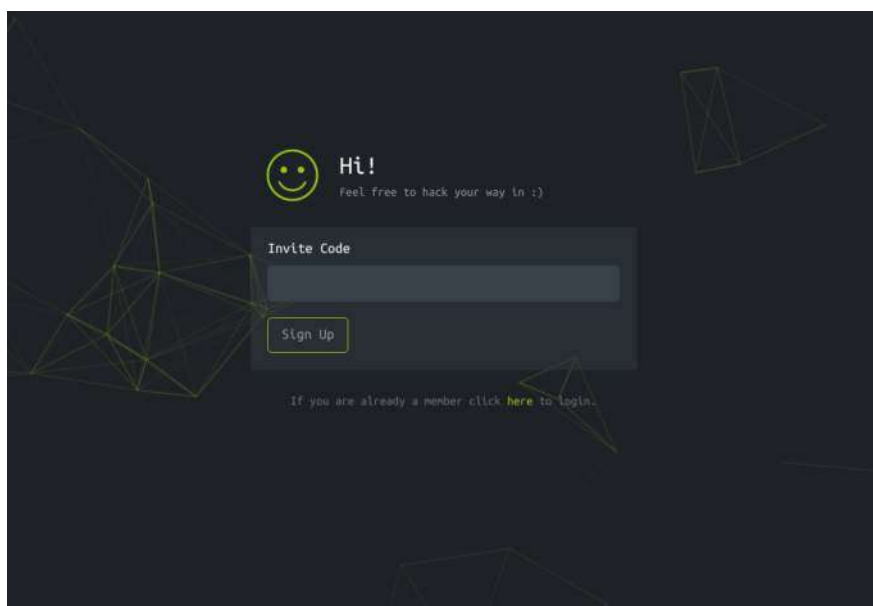


Figura 5.1: Schermata per la registrazione sulla piattaforma Hackthebox.

5.2 Accesso con Invite Code

Ho seguito diversi approcci al problema, basandomi sulle conoscenze acquisite.

5.2.1 Tentativo 1: Injection

Per accertarmi che il campo “invite code” non fosse vulnerabile a SQL Injections ed a XXE Attacks ho intercettato il traffico in uscita con Burp e ho modificato le richieste HTTP, l’unico risultato che ho ottenuto è stato la schermata 5.2.

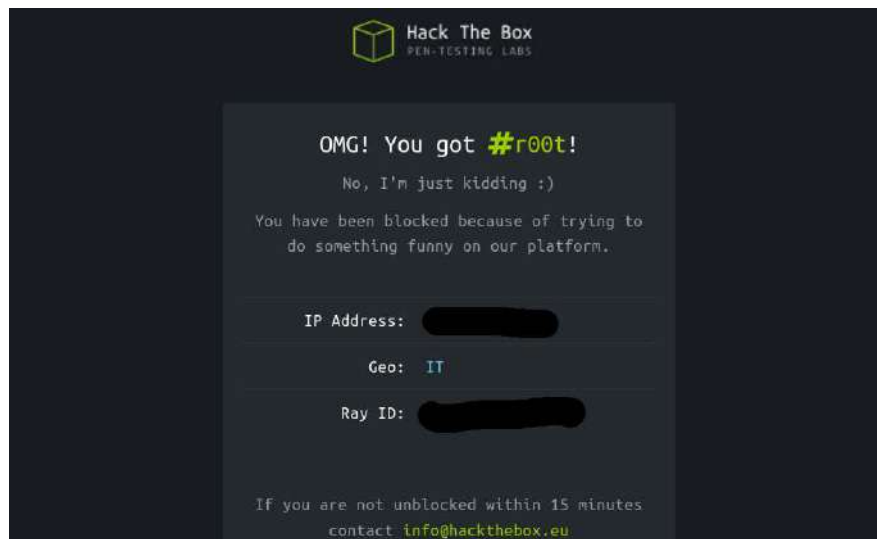


Figura 5.2: Risposta a tentativi di injection.

... sì, hanno il senso dell’umorismo.

Non ha funzionato nemmeno modificando i caratteri speciali con l’equivalente combinazione ASCII.

In una situazione reale non sarebbe stato così chiaro che questa non fosse la strada da seguire, ma per questa competizione esiste necessariamente una soluzione, che non è questa.

5.2.2 Tentativo 2: Token analysis

Una possibilità che mi era venuta in mente era la decodifica del token intercettato tramite Burp. Esso solitamente ha codifica in base 64, ed effettivamente risulta essere una porzione di JSON.



Figura 5.3: Invio dell'invite code ad Hackthebox intercettato da Burp.

Il JSON è il seguente:

```
{"iv":"IF1[...]0g==","value":"iN[...]==" ,"mac":"e5[...]19"}
```

Che sembrano essere le informazioni relative ad uno scambio di informazioni Cypher Block Chaining, quindi è inutile proseguire per questa strada... Non sembra essere quella giusta.

5.2.3 Tentativo 3: Code analysis

Nel sorgente JavaScript, cercando nel codice html, si trova una porzione di codice che sembra essere responsabile della valutazione dell'invite code. Eseguendo lo script vediamo che questo codice:

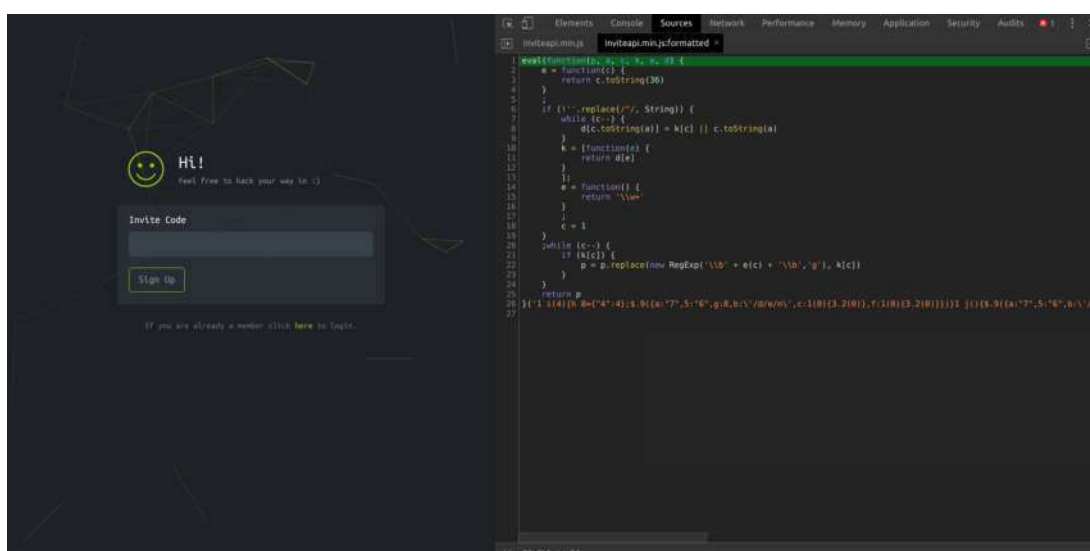


Figura 5.4: Javascript trovato nelle risorse della pagina.

è equivalente a questo:

```

1 function verifyInviteCode(code) {
2     var formData = {
3         "code": code
4     };
5     $.ajax({
6         type: "POST",
7         dataType: "json",
8         data: formData,
9         url: '/api/invite/verify',
10        success: function(response) {
11            console.log(response)
12        },
13        error: function(response) {
14            console.log(response)
15        }
16    })
17 }
18
19 function makeInviteCode() {
20     $.ajax({
21         type: "POST",
22         dataType: "json",
23         url: '/api/invite/how/to/generate',
24         success: function(response) {
25             console.log(response)
26         },
27         error: function(response) {
28             console.log(response)
29         }
30     })
31 }

```

Figura 5.5: Generated code from 5.4.

Il quale ci fornisce un'informazione importante: che esiste un modo per generare un Invite Code. Quindi cerchiamo di chiamare la funzione `makeinviteCode()` dalla console, ottenendo così la risposta:

```
{data: "Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFg
erdhrfg gb /ncv/vaivgr/trarengr", enctype: "ROT13"}
```

Decifriamo <https://rot13.com/> online il campo data con enctype:

In order to generate the invite code, make a POST request to
</api/invite/generate>

Sempre da console, è possibile generare una richiesta post molto semplicemente tramite l'oggetto `XMLHttpRequest`.

Lo screenshot della console (figura 5.6) racchiude tutti i comandi eseguiti.

```

> makeInviteCode
< f makeInviteCode()
  {$.ajax({type:"POST",dataType:"json",url:'/api/invite/how/to/generate',success:function(response)
  {console.log(response)},error:function(response){console.log(response)}})}
> makeInviteCode()
< undefined
  ▶ {0: 200, success: 1, data: {...}} VM33:1
> xhr = new XMLHttpRequest()
< XMLHttpRequest {onreadystatechange: null, readyState: 0, timeout: 0, withCredentials: false, upload: XMLHttpRequest
  ▶ stUpload, ...}
> url = "/api/invite/generate"
< "/api/invite/generate"
> xhr.open("POST",url)
< undefined
> xhr.send()
< undefined
> xhr.response
< '{"success":1,"data":{"code":"V0ZFSVct5LlDTl0tREp0VKtENhEWUitVlZCWlg=","format":"encoded"},"0":200}'
>

```

Figura 5.6: Screenshot della console js di hackthebox.

Rimane solo da decodificare il campo “code” dal formato base64 a plain text.

WFEIW-JYCNT-DJCVA-XCDYB-VVBZX

Questa era la strada giusta da percorrere per ottenere l’invite code. Ora si dovrebbe iniziare qualche challenge interna alla piattaforma!

5.3 Challenge #1: Netmon

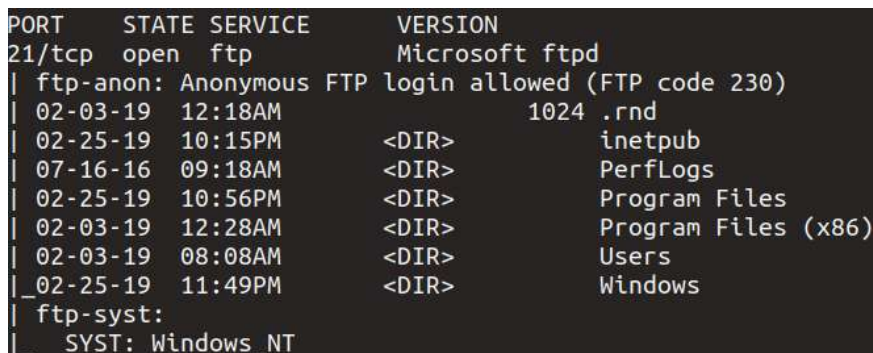
Questa challenge consisteva nel prendere permessi utente e permessi di root senza credenziali di accesso. Possedendo solo l’indirizzo IP della macchina, ho lanciato il comando nmap per ottenere una panoramica dei servizi in ascolto sulle porte del server.

5.3.1 User Access

Dall’analisi di nmap si capisce che sulla porta 21 è disponibile un accesso ftp anonimo, quindi con il comando bash:

```
$ ftp 10.10.10.152 21
```

e inserendo le credenziali *anonymous:anonymous*, è possibile accedere al filesystem (figura 5.7).



```
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 02-03-19 12:18AM          1024 .rnd
| 02-25-19 10:15PM          <DIR>  inetpub
| 07-16-16 09:18AM          <DIR>  PerfLogs
| 02-25-19 10:56PM          <DIR>  Program Files
| 02-03-19 12:28AM          <DIR>  Program Files (x86)
| 02-03-19 08:08AM          <DIR>  Users
|_ 02-25-19 11:49PM          <DIR>  Windows
| ftp-syst:
|_ SYST: Windows_NT
```

Figura 5.7: Root directory of the server.

Quindi, è possibile effettuare un login anonimo per ottenere il contenuto del primo file richiesto dalla sfida, che si trova sul desktop dell’utente generico.

Accedendo con Anonymous è possibile scaricare il file user.txt, contenente il flag che ci interessava per validare la prima parte della sfida.

5.3.2 Root Access

Questa parte della sfida è più complicata. Per capire come ottenere le credenziali di accesso di un utente con permessi di root è stato necessario indagare sul network

manager. Da come si può vedere dalla pagina iniziale del sito, il sistema è gestito dal PRTG Network Monitor, e per di più da una versione gratuita.

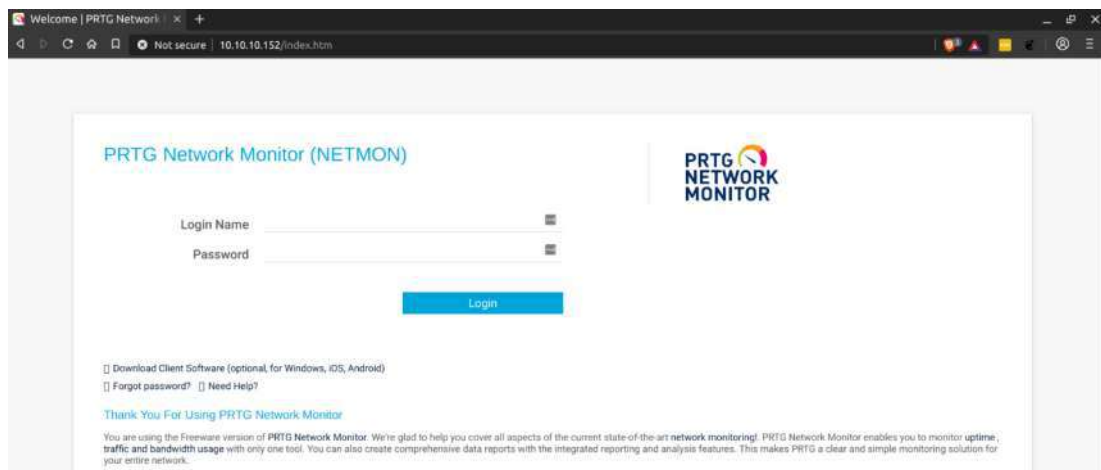


Figura 5.8: Login page of Netmon server.

Con una veloce ricerca [13], si viene subito a conoscenza del fatto che una versione obsoleta di questo manager salva in chiaro le credenziali di accesso, ma anche se questo fosse aggiornato, potrebbero esserci ancora delle tracce nei backup dei file di configurazione. Infatti, cercando nel percorso:

C:/ProgramData/Paessler/PRTG Network Monitor

si trova il file “PRTG Configuration.dat.old” al cui interno ci sono le credenziali.

```
134      <dbauth>
135      0
136      </dbauth>
137      <dbcredentials>
138      0
139      </dbcredentials>
140      <dbpassword>
141      <!-- User: prtgadmin -->
142      PrTg@dmin2019
143      </dbpassword>
144      <dbtimeout>
145      60
146      </dbtimeout>
147      <depdelay>
148      0
149      </depdelay>
150      <dependencytype>
151      0
152      </dependencytype>
153      <discoveryschedule>
```

Figura 5.9: Credenziali salvate in un file di configurazione.

Grazie a queste è possibile accedere alla pagina web di monitoraggio come admin. Dalla schermata è possibile far eseguire un comando specifico al server, ed io ho scelto di caricare un file .bat in grado di eseguire una copia del file contenente il secondo flag in una cartella accessibile con privilegi da user. Accedendo con la stessa modalità vista nella prima parte, si accede alla cartella scelta e si scarica il file contenente il flag. **Sfida Completata!**

Capitolo 6

Piattaforma di pratica: Root-me

6.1 Descrizione

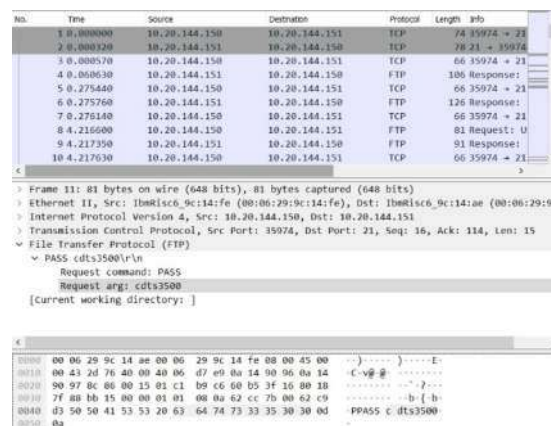
Questa piattaforma di pratica, disponibile all'indirizzo <https://www.root-me.org>, mette a disposizione diverse challenge su diverse categorie. Io affronto qui quelle di networking.

6.2 Challenge #1: FTP Authentication

“An authenticated file exchange achieved through FTP. Recover the password used by the user.”

L'obiettivo è analizzare un file di traffico di rete per ottenere informazioni riguardo la password di un utente che vi è connesso con protocollo ftp.

Il file è analizzabile facilmente tramite Wireshark [\[download link\]](#) (figura 6.1 e con la conoscenza dell'RFC 959 [\[7\]](#).



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.20.144.150	10.20.144.151	TCP	74	45974 → 21
2	0.000320	10.20.144.151	10.20.144.150	TCP	78	21 → 35974
3	0.000570	10.20.144.150	10.20.144.151	TCP	66	35974 → 21
4	0.000630	10.20.144.151	10.20.144.150	FTP	106	Response: 21
5	0.275440	10.20.144.150	10.20.144.151	TCP	66	35974 → 21
6	0.275760	10.20.144.151	10.20.144.150	FTP	126	Response: 21
7	0.276100	10.20.144.150	10.20.144.151	TCP	66	35974 → 21
8	4.216600	10.20.144.150	10.20.144.151	FTP	81	Request: U
9	4.217350	10.20.144.151	10.20.144.150	FTP	81	Response: U
10	4.217630	10.20.144.150	10.20.144.151	TCP	66	35974 → 21

Frame 11: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface 0	
Ethernet II, Src: Intel(R) Ethernet Controller (08:00:27:9C:14:FE), Dst: Intel(R) Ethernet Controller (08:00:27:9C:14:FE)	
Internet Protocol Version 4, Src: 10.20.144.150, Dst: 10.20.144.151	
Transmission Control Protocol, Src Port: 35974, Dst Port: 21, Seq: 16, Ack: 114, Len: 15	
File Transfer Protocol (FTP)	
Request command: PASS	
Request arg: cdt3500	
[Current working directory:]	

Raw Data	
0000	00 06 29 9c 14 fe 00 06 29 9c 14 fe 00 00 45 00
0010	00 43 2d 76 40 19 01 c1 b9 c6 60 b5 3f 16 80 1b
0020	90 97 8c 86 00 15 01 c1 b9 c6 60 b5 3f 16 80 1b
0030	7f 88 bb 15 00 00 01 01 00 0a 62 cc 7b 00 62 c9
0040	d3 50 50 41 53 53 20 63 64 74 73 33 35 30 30 0d
0050	0a

Figura 6.1: Dump file analyzed with Wireshark.

6.3 Challenge #2: TELNET Authentication

“Find the user password in this TELNET session capture.”

L’obiettivo è analizzare un file di traffico di rete per ottenere informazioni riguardo la password di un utente che vi è connesso con protocollo telnet. Il file è analizzabile facilmente tramite WireShark [\[download link\]](#) e con la conoscenza dell’RFC 854 [\[6\]](#).

6.4 Challenge #3: ETHERNET Frame

“Find the (supposed to be) confidential data in this ethernet frame:”

L’obiettivo è analizzare un frame Ethernet per ricavare dei dati confidenziali al loro interno. Il frame è:

```
00 05 73 a0 00 00 e0 69 95 d8 5a 13 86 dd 60 00 00 00 00 9b 06 40 26 07 53 00 00 60 2a bc 00
00 00 00 ba de c0 de 20 01 41 d0 00 02 42 33 00 00 00 00 00 00 00 04 96 74 00 50 bc ea 7d b8
00 c1 d7 03 80 18 00 e1 cf a0 00 00 01 01 08 0a 09 3e 69 b9 17 a1 7e d3 47 45 54 20 2f 20 48
54 54 50 2f 31 2e 31 0d 0a 41 75 74 68 6f 72 69 7a 61 74 69 6f 6e 3a 20 42 61 73 69 63 20 59 32
39 75 5a 6d 6b 36 5a 47 56 75 64 47 6c 68 62 41 3d 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20
49 6e 73 61 6e 65 42 72 6f 77 73 65 72 0d 0a 48 6f 73 74 3a 20 77 77 77 2e 6d 79 69 70 76 36
2e 6f 72 67 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a 0d 0a
```

Figura 6.2: Frame Ethernet catturato (esadecimale).

E con un po’ di studio dell’RFC 1042 [\[5\]](#) si risolve facilmente decodificando dall’esadecimale in testo:

```
GET / HTTP/1.1
Authorization: Basic Y29uZmk6ZGVudGhhbA==
User-Agent: InsaneBrowser
Host: www.myipv6.org
Accept: */*
```

Da cui vediamo che la parte di autorizzazione contiene una stringa codificata in base64, che tradotta ci fornisce le credenziali:

```
confi:dential
```

Parte III

L'azienda e l'attività

Capitolo 7

Webinar: Frida Framework

7.1 Introduzione

Nell'azienda vengono organizzati spesso Webinar e Bootlabs tenuti da altri dipendenti Reply in tutta Europa. Sono molto tecnici e specifici, ma io ho deciso di partecipare a questo.

7.2 What is FRIDA?

Dal momento che le applicazioni con cui veniamo a contatto sono spesso già compilate. Questo toolkit open source serve ad interagire con programmi (running) di cui non abbiamo il codice sorgente. Frida funziona come un server che viene eseguito da un dispositivo.

7.3 What is it for?

Può essere usato efficacemente per analizzare il comportamento di un'applicazione: per esempio è possibile vedere che genere di crittografia utilizza l'applicazione con cui comunica, oppure attivare la modalità di debug (che magari è ancora codificata nell'applicazione ma non accessibile direttamente). Insomma è uno strumento di analisi [\[GitHub link\]](#).

7.4 Installation and Basic Usage

Per OS Linux/Unix:

```
$ sudo pip install frida-tools
```

7.5 Frida Cookbook

Frida usa Javascript per istanziare classi Objective C per iOS e Java per Android.

```
1 // Segno solo Android, anche se è più difficile
2 Java.perform( function (){
3     const JavaString =
4         Java.use('java.lang.String');
5
6     var exampleString = JavaString.$new('Hello World!');
7 })
```

Figura 7.1: Istanziamento di un oggetto "stringa" in un applicativo con FRIDA.

Frida può essere uno strumento molto interessante se usato propriamente, è utile e enfatizzato per l'utilizzo con mobile apps. Non rientra nell'argomento del mio tirocinio, ma ho imparato qualcosa. [\[Website link\]](#)

Capitolo 8

Reportistica

8.1 Common Vulnerability Scoring System

Il CVSS [15] è uno standard aperto che serve per stabilire la gravità di una vulnerabilità nella sicurezza di un sistema informatico.

8.2 CVSS2 Score

Il CVSS2 Score [2] è uno standard utilizzato per descrivere le caratteristiche e l'impatto di una vulnerabilità. Si compone di tre indici primari: “Base vector”, “Temporal vector” ed “Environment vector”: ogni indice è composto da un riferimento numerico, variabile tra 0 e 10, che identifica la gravità della vulnerabilità e da un vettore che ne specifica le caratteristiche (per una dettagliata documentazione relativa il calcolo e le specifiche dello standard CVSS2 e CVSS3, si faccia riferimento al NVD Common Vulnerability Scoring System [12]).

8.3 Metrica di base

Il gruppo di metriche di base identifica le caratteristiche costanti nel tempo di una vulnerabilità. L'Access Vector, l'Access Complexity e l'Authentication sono metriche che permettono di definire le modalità con cui è possibile sfruttare una vulnerabilità, e le necessarie condizioni alfine di effettuare un attacco con successo. Le tre metriche misurano in che modo la vulnerabilità, se sfruttata, influirà su una risorsa IT, in cui gli impatti sono definiti in termini di perdita di riservatezza, integrità e disponibilità.

8.4 Access Vector (AV)

Questa metrica definisce la modalità tramite cui è possibile sfruttare le vulnerabilità. I possibili valori sono per questo tipo di metrica sono riportati nella tabella seguente.

Valore	Descrizione
Local (L)	La vulnerabilità è sfruttabile unicamente avendo un accesso al target con privilegi locali; l'aggressore deve quindi avere un accesso fisico al target o una shell remota disponibile.
Adjacent Network (A)	La vulnerabilità è sfruttabile unicamente se l'aggressore ha accesso ad una rete collegata a quella dov'è presente il target (IP subnet, Bluetooth, IEEE 802.11).
Network (N)	La vulnerabilità è sfruttabile senza avere accesso alla rete locale; in genere, queste vulnerabilità sono dette "remotely exploited" (es. RPC buffer overflow)

8.5 Access Complexity (AC)

Questa metrica descrive la complessità necessaria ad eseguire l'attacco: alcune vulnerabilità, per poter essere sfruttate con successo, richiedono diverse azioni quali, ad esempio, l'apertura di un file infetto o lo scaricamento di un allegato da e-mail.

Valore	Descrizione
High (H)	Esistono condizioni specializzate di accesso. Per esempio: <ul style="list-style-type: none">• Nella maggior parte delle configurazioni, la parte attaccante deve avere già privilegi elevati o effettuare spoofing di sistemi aggiuntivi oltre al sistema di attacco (ad esempio, DNS hijacking).• L'attacco dipende da metodi di ingegneria sociale che potrebbero essere facilmente rilevati da persone ben informate. Ad esempio, la vittima deve eseguire diverse azioni sospette o atipiche.

Medium (M)	<p>Le condizioni di accesso sono in qualche modo specializzate. Per esempio:</p> <ul style="list-style-type: none"> • La parte attaccante è limitata a un gruppo di sistemi o utenti ad un certo livello di autorizzazione, probabilmente non attendibile. • Alcune informazioni devono essere raccolte prima che possa essere lanciato un attacco che vada a buon fine. • La configurazione interessata non è predefinita e comunemente non è configurata (ad esempio, vulnerabilità presente quando un server esegue l'autenticazione dell'account utente tramite uno schema specifico, ma non è presente per un altro schema di autenticazione). • L'attacco richiede una piccola quantità di ingegneria sociale che potrebbe occasionalmente ingannare gli utenti cauti (ad esempio, attacchi di phishing che modificano la barra di stato di un browser Web per mostrare un falso collegamento, dovendo essere nell'elenco "amico" di qualcuno prima di inviare un IM exploit).
Low (L)	<p>Non esistono condizioni specializzate di accesso o circostanze attenuanti. Per esempio:</p> <ul style="list-style-type: none"> • Il prodotto interessato in genere richiede l'accesso a una vasta gamma di sistemi e utenti, possibilmente anonimi e non attendibili (ad es., Internet o server di posta Internet). • La configurazione interessata è predefinita o ubiquitaria. • L'attacco può essere eseguito manualmente e richiede poca abilità o raccolta di informazioni aggiuntive. • La "race condition" è pigra (cioè, è tecnicamente una race ma facilmente vincibile).

8.6 Authentication (Au)

Questa metrica misura il numero di volte che un utente malintenzionato deve autenticarsi su un target per sfruttare la vulnerabilità. Questa metrica non misura il tipo o la complessità del processo di autenticazione, ma prende in considerazione solo se ad un utente malintenzionato è richiesto di fornire le credenziali prima che si verifichi un exploit. I possibili valori per questa metrica sono elencati nella tabella seguente. Minore è il numero di istanze di autenticazione richieste, maggiore è il punteggio di vulnerabilità.

Valore	Descrizione
Multiple (M)	Sfruttare la vulnerabilità richiede che l'attaccante effettui l'autenticazione due o più volte, anche se vengono utilizzate le stesse credenziali ogni volta
Single (S)	È richiesta un'istanza di autenticazione per accedere e sfruttare la vulnerabilità.
None (N)	L'autenticazione non è richiesta per accedere e sfruttare la vulnerabilità.

8.7 Confidential Impact (C)

Questa metrica misura l'impatto sulla riservatezza di una vulnerabilità sfruttata con successo. La riservatezza si riferisce alla limitazione dell'accesso e della divulgazione delle informazioni solo agli utenti autorizzati, nonché alla prevenzione dell'accesso o della divulgazione a persone non autorizzate. I possibili valori per questa metrica sono elencati nella tabella seguente. Un maggiore impatto sulla riservatezza aumenta il punteggio di vulnerabilità.

Valore	Descrizione
None (N)	Non vi è alcun impatto sulla riservatezza del sistema.
Partial (P)	Vi è una notevole divulgazione di informazioni. L'accesso ad alcuni file di sistema è possibile, ma l'attaccante non ha il controllo su ciò che viene ottenuto, o l'ambito della perdita è limitato.
Complete (C)	Vi è una divulgazione totale delle informazioni, con la conseguente rivelazione di tutti i file di sistema. L'autore dell'attacco è in grado di leggere tutti i dati del sistema (memoria, file, ecc.).

8.8 Integrity Impact (I)

Questa metrica misura l'impatto sull'integrità di una vulnerabilità sfruttata con successo. L'integrità si riferisce all'affidabilità e alla veridicità garantita delle informazioni. I possibili valori per questa metrica sono elencati nella tabella seguente. Un maggiore impatto sull'integrità aumenta il punteggio di vulnerabilità.

Valore	Descrizione
None (N)	Non vi è alcun impatto sulla disponibilità del sistema.
Partial (P)	Vi sono prestazioni ridotte o interruzioni nella disponibilità delle risorse. Un esempio è un attacco di tipo "flood" basato sulla rete che consente un numero limitato di connessioni riuscite a un servizio Internet.
Complete (C)	C'è un arresto totale della risorsa interessata. L'autore dell'attacco può rendere la risorsa completamente non disponibile.

8.9 Availability Impact (A)

Questa metrica misura l'impatto sulla disponibilità di una vulnerabilità sfruttata con successo. La disponibilità si riferisce all'accessibilità delle risorse informative. Gli attacchi che consumano la larghezza di banda della rete, i cicli del processore o lo spazio su disco hanno un impatto sulla disponibilità di un sistema. I possibili valori per questa metrica sono elencati nella tabella seguente. L'aumento dell'impatto sulla disponibilità aumenta il punteggio di vulnerabilità.

Valore	Descrizione
None (N)	Non vi è alcun impatto sulla disponibilità del sistema.
Partial (P)	Vi sono prestazioni ridotte o interruzioni nella disponibilità delle risorse. Un esempio è un attacco di tipo "flood" basato sulla rete che consente un numero limitato di connessioni riuscite a un servizio Internet.
Complete (C)	C'è un arresto totale della risorsa interessata. L'autore dell'attacco può rendere la risorsa completamente non disponibile.

8.10 Livelli di Severità

Il punteggio CVSSv2, riportato sopra, verrà utilizzato nel presente documento per assegnare un livello di severity a ciascun problema di sicurezza. La tabella seguente riassume i possibili valori di criticità con una breve descrizione del loro significato.

Valore	Descrizione
Critical	La vulnerabilità consente all'autore dell'attacco di compromettere completamente l'applicazione di destinazione. (Punteggio CVSS: 10,0).
High	La vulnerabilità consente l'esecuzione di codice dannoso, accesso non autorizzato alle sezioni amministrative dell'applicazione, per accedere ai dati memorizzati nei database. (Punteggio CVSS: 7.0-9.9).
Medium	La vulnerabilità consente di acquisire informazioni sensibili o comunque importanti che possono essere utilizzate negli attacchi successivi. La vulnerabilità potrebbe consentire a un utente malintenzionato di modificare il normale funzionamento dell'applicazione. (Punteggio CVSS: 4.0-6.9).
Low	La vulnerabilità consente di acquisire informazioni sulla configurazione o almeno consentire un attacco con un impatto limitato sull'attività del cliente. (Punteggio CVSS: 0,0-3,9).

8.11 Esempio

Ad esempio, possiamo considerare le seguenti vulnerabilità con i relativi dettagli:

- **Access Vector:** Low;
- **Access Complexity:** Medium;
- **Authentication:** None;
- **Confidentiality Impact:** None;
- **Integrity Impact:** Partial;
- **Availability Impact:** Complete;

E il vettore associato è: **AV:L/AC:M/Au:N/C:N/I:P/A:C**.

8.12 Riferimenti

Per ogni vulnerabilità rilevata viene indicato, quando possibile, un riferimento a uno o più database di vulnerabilità facilmente accessibili in Internet. Questi riferimenti consentono di recuperare, nei siti elencati, ulteriori informazioni sulla vulnerabilità segnalata.

Riferimento	Descrizione	URL
CVE	Common Vulnerabilities and Exposures	http://www.cve.mitre.org
CAN	CVE Candidate Name	http://www.cve.mitre.org
BID	Bugtraq ID	http://www.securityfocus.com/bid
XF	ISS XForce Database	http://xforce.iss.net/
SECUNIA	Secunia Bulletin	http://secunia.com
OSVDB	Open Source Vulnerability Data Base	http://www.ovsdb.org
OWASP	Open Web Application Security Project	http://www.owasp.org
RHSA	Red Hat Security Advisory	http://www.redhat.com
CIAC	Computer Incident Advisory Capability	http://ciac.llnl.gov/ciac/index.html
MS	Microsoft Security Bulletin	http://security.microsoft.com
CERT	CERT Vulnerability Database	http://www.cert.org/kb/
NESSUS	Nessus ID	http://www.nessus.org

Capitolo 9

Attività Reply

9.1 Introduzione

Security Reply è azienda leader nel settore della consulenza per la security ed è stato importante per me avvicinarmi a questo ambito osservando proprio questa azienda.

Tutto ciò che è contenuto nella relazione è frutto di studio personale, sostenuto e integrato dall'esperienza di coloro con cui sono venuto a contatto nell'ambiente di lavoro. Non sono argomenti che mi sarebbe stato possibile apprendere altrimenti con questa qualità, in modo autonomo e in così poco tempo.

9.2 Organizzazione e struttura

Come accennato nell'introduzione nel capitolo 1, sono stato messo nell'open space, dove lavorano sia junior che senior consultants, mentre i manager che gestiscono le 3 diverse Business Units hanno spazi dedicati.

I criteri di sicurezza e privacy che si devono seguire sono molto precisi e coinvolgono sia gli accordi di non divulgazione tra clienti e Reply, sia l'attenzione per la gestione delle utenze e dei premissi nelle operazioni reali sulle piattaforme di testing. Per i nuovi assunti è necessario, entro un tempo limite, completare una serie di corsi, a fruizione digitale, riguardanti privacy, GDPR e sicurezza sul lavoro.

9.3 Nuove competenze acquisite

Nel periodo di tirocinio ho acquisito le basi minime che mi permetterebbero di lavorare in quest'ambito sin da subito come apprendista, facendolo ho avuto

l'occasione di scoprire, osservare e vivere, in parte, l'atmosfera dell'azienda, con le sue scadenze, i suoi obblighi e le sue relazioni tra colleghi.

Per quanto concerne la mia formazione, l'esperienza è servita per acquisire le conoscenze fondamentali nell'ambito della sicurezza delle web applications, inquadrata nel contesto moderno e imperfetto delle piattaforme reali.

Come visto nei capitoli precedenti, ho acquisito competenze in diversi argomenti:

- **OWASP Top 10 e contromisure** (Capitoli 3 e 4): ho imparato le più comuni vulnerabilità delle WA mettendone in pratica l'efficacia in ambiente di test predisposto (WebGoat), chiedendo approfondimenti e chiarimenti ai consulenti da cui ero circondato.
- **Situazioni Pseudo-Reali** (Capitoli 5 e 6): Le conoscenze e l'aiuto che ho ricevuto mi hanno permesso di portare a termine qualche sfida predisposta da famose piattaforme di allenamento.
- **Reportistica e modello di comunicazione** (Capitolo 8): ho imparato come deve essere comunicata la presenza di una vulnerabilità seguendo degli standard precisi.
- **Situazioni Reali**: non potendo parlare estensivamente non ho approfondito molto questo aspetto, ma ho avuto modo di partecipare ad alcune attività di analisi dinamica e statica per le piattaforme web dei clienti di Security Reply. Nonostante non abbia mai agito da solo, ho contribuito il più possibile in queste attività.

9.4 Conclusione

Ho avuto modo di scoprire come dei consulenti tecnici esperti svolgono il proprio lavoro nella security, degli strumenti e degli approcci che usano, dei rapporti con i clienti e con i manager.

Dopo questa esperienza rimango convinto del fatto che la security sia la mia passione e la strada su cui sono genuinamente convinto di proseguire il mio cammino.

Bibliografia

- [1] Kali Docs. *Making a Kali Bootable USB Drive*. Constantly updated. URL: <https://docs.kali.org/downloading/kali-linux-live-usb-install>.
- [2] FIRST. *Common Vulnerability Scoring System v2*. Consultato il 9 apr 2019. URL: <https://www.first.org/cvss/v2/guide/>.
- [3] Gartner Glossary. *Static Application Security Testing (SAST)*. Consultato il 9 apr 2019. URL: <https://www.gartner.com/it-glossary/static-application-security-testing-sast>.
- [4] Henry Hoggard. *Paypal: 2-Factor-Authentication Bypass*. 2016. URL: <http://henryhoggard.co.uk/blog/Paypal-2FA-Bypass>.
- [5] IETF. *RFC 1042*. 1988. URL: <https://www.rfc-editor.org/info/rfc1042>.
- [6] IETF. *RFC 854*. 1983. URL: <https://www.rfc-editor.org/info/rfc854>.
- [7] IETF. *RFC 959*. 1985. URL: <https://www.rfc-editor.org/info/rfc959>.
- [8] Stefano Novelli. *Hacklog Vol.2*. inforge.net, 2018.
- [9] Stack Overflow. *Comment on REST paradigm*. 2009 - currently updated. URL: <https://stackoverflow.com/a/671132>.
- [10] OWASP. *OWASP top 10*. owasp.org, 2017.
- [11] OWASP. *WebGoat Project*. Constantly updated. URL: https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project.
- [12] Sasha Romanosky Peter Mell Karen Scarfone. «The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems». In: *public available PDF* 1.NIST Interagency Report 7435 (2007), p. 33.
- [13] Reddit. *PRTG gave away some of your passwords*. 2018. URL: https://www.reddit.com/r/sysadmin/comments/862b8s/prtg_gave_away_some_of_your_passwords/.
- [14] Wikipedia. *Billion laughs attack*. Constantly updated. URL: <https://en.wikipedia.org/wiki/BillionLaughsAttack>.

- [15] Wikipedia. *Common Vulnerability Scoring System*. Constantly updated. URL: https://en.wikipedia.org/wiki/Common_Vulnerability_Scoring_System.
- [16] Wikipedia. *Document Object Model*. Constantly updated. URL: https://en.wikipedia.org/wiki/Document_Object_Model.
- [17] Wikipedia. *Man in the middle*. Constantly updated. URL: https://en.wikipedia.org/wiki/Man-in-the-middle_attack.
- [18] Wikipedia. *Object Relational Mapping*. Constantly updated. URL: https://en.wikipedia.org/wiki/Object-relational_mapping.