

Università degli Studi di Torino

Dipartimento di informatica



Tesi di Laurea Triennale in Informatica

Analisi di vulnerabilità all'interno di Web Application e Cloud

Relatore:

Idilio Drago

Candidato:

Ilaria Sena

ANNO ACCADEMICO 2022/2023

"L'istruzione è l'arma più potente che puoi usare per cambiare il mondo."
- Nelson Mandela

Abstract

Le vulnerabilità presenti all'interno di applicazioni possono costituire un rischio significativo, suscettibile di attacchi da parte di diverse entità, i quali possono compromettere dati sensibili, interrompere servizi critici e causare danni notevoli. Pertanto, è cruciale che ogni entità operante nel campo dell'informatica, comprese le aziende non specializzate in sicurezza informatica, adotti le misure preventive appropriate.

Blue Reply, un'azienda consulenziale focalizzata sullo sviluppo di applicazioni web, è un'azienda che si impegna nell'implementare pratiche di programmazione sicura al fine di garantire una maggiore sicurezza nei loro prodotti e soddisfare le esigenze dei propri clienti.

La mia ricerca condotta presso Blue Reply sarà di natura prevalentemente teorica e si articolerà in quattro diverse fasi. Inizialmente, verranno esposte le motivazioni che rendono cruciale l'analisi delle vulnerabilità in ambito web a partire dall'innovazione digitale avvenuta nel 2020 a causa della pandemia, evidenziando l'urgenza del problema attuale tramite diversi grafici e dati numerici.

Successivamente, l'attenzione sarà rivolta alle principali minacce note che possono riscontrarsi su applicazioni web, prenderò in considerazione le classificazioni e gli studi più recenti, analizzando le minacce ed alcune delle loro possibili risoluzioni; verranno inoltre anche indicati alcuni degli attacchi più recenti tramite le minacce prese in analisi.

Dopo di che si analizzeranno nel dettaglio le metodologie di valutazione delle vulnerabilità e le più importanti best practice da adottare per garantire una programmazione sicura. In questa fase verranno illustrate le tecnologie e le applicazioni più utilizzate e verranno anche inclusi esempi veri e propri di valutazione del rischio in azienda per le applicazioni web.

Come ultima fase del mio studio ho scelto di trattare le vulnerabilità delle applicazioni Cloud, argomento molto attuale con il quale l'azienda Blue Reply si sta iniziando ad interfacciare; verranno illustrate le vulnerabilità e le best practice più comuni.

L'obiettivo di questa tesi è quindi quello apprendere i rischi legati ad una programmazione poco attenta sapendo riconoscere e correggere le vulnerabilità più note, inoltre si vuole fornire un'analisi delle procedure fondamentali per identificare e valutare le vulnerabilità presenti all'interno di applicazioni web e cloud, offrendo un contributo significativo al campo della sicurezza informatica dell'azienda.

Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata.

Indice

1	Introduzione e motivazioni dell'importanza della sicurezza	7
1.1	Il processo di innovazione digitale	7
1.2	Lo sviluppo del rischio	7
1.3	Situazioni di rischio attuali	8
1.4	Analisi cyber attacchi a livello globale	8
2	Minacce web note e risoluzioni	11
2.1	Il concetto di malware	11
2.1.1	Come proteggersi dai malware	12
2.1.2	Attacchi Malware recenti	13
2.2	Attacchi DDoS	13
2.2.1	Attacchi DDoS recenti	13
2.3	Attacchi zero-day	13
2.3.1	Attacchi zero-day recenti	14
2.4	OWASP : Vulnerabilità comuni e classificazioni	14
2.4.1	Ultimi cambiamenti Owasp Top 10	14
2.4.2	Broken Access Control	15
2.4.3	Prevenire Broken Access Control	15
2.4.4	Cryptographic Failures	15
2.4.5	Prevenire Cryptographic Failures	16
2.4.6	Injection	16
2.4.7	Prevenire Injection	16
2.4.8	Insecure Design	17
2.4.9	Prevenire Insecure Design	17
2.4.10	Security Misconfiguration	17
2.4.11	Prevenire Security Misconfiguration	18
2.4.12	Vulnerable and Outdated Components	18
2.4.13	Prevenire Vulnerable and Outdated Components	18
2.4.14	Identification and Authentication Failures	19
2.4.15	Prevenire Identification and Authentication Failures	19
2.4.16	Software and Data Integrity Failures	21
2.4.17	Prevenire Software and Data Integrity Failures	21
2.4.18	Security Logging and Monitoring Failures	21
2.4.19	Prevenire Security Logging and Monitoring Failure	21
2.4.20	Server-Side Request Forgery	22
2.4.21	Prevenire Server-Side Request Forgery	22

3	La valutazione del rischio digitale per applicazioni web	23
3.1	Risk assessment e risk management	23
3.2	Integrazione continua DevSecOps	25
3.2.1	Vantaggi DevSecOps	25
3.2.2	Pipeline DevSecOps	25
3.3	Analisi dei report	26
3.4	Tecnologie DevSecOps	26
3.4.1	Jenkins	27
3.4.2	Snyk	28
3.4.3	OWASP Dependency-Check	29
3.4.4	Owasp ZAP	29
3.4.5	Esempi pratici	30
4	Minacce Cloud note e risoluzioni	37
4.0.1	Vulnerabilità comuni in applicazioni Cloud	37
4.0.2	Pratiche di gestione delle vulnerabilità del Cloud	38
5	Conclusioni finali	39

Elenco delle figure

1.1	Attacchi per semestre	9
1.2	Attacchi mensili	9
2.1	Cambiamenti top10	14
3.1	Valutazione del rischio	23
3.2	Pipeline di successo	26
3.3	Pipeline di fallimento	28
3.4	Snyk Vulnerabilities	28
3.5	Applicazione ZAP	29
3.6	ZAP report	30
3.7	Avvio Virtual machine NotSecureBank	31
3.8	Virtual machine NotSecureBank	31
3.9	Scanner Zap	32
3.10	Report finale NotSecureBank	32
3.11	Installazione npm	33
3.12	npm audit	34
3.13	Visualizzazione di vulnerabilità	34
3.14	Visualizzazione applicazione in Docker	35
3.15	Applicazione HackMEAN	35
3.16	Scanner HackMEAN	36
3.17	Report HackMEAN	36

Capitolo 1

Introduzione e motivazioni dell'importanza della sicurezza

1.1 Il processo di innovazione digitale

Il processo di digitalizzazione ha subito un'accelerazione grazie alla pandemia da Covid-19 che ha accresciuto ed accelerato l'esigenza di adottare applicazioni, infrastrutture e dotazioni tecnologiche. La nuova normalità mira quindi ad un lavoro sempre più agile con l'obiettivo di semplificare ed innovare i processi di funzionamento e di garantire un sistema più efficiente ed efficace. Il digitale diviene un elemento imprescindibile al fine di rendere più efficienti i processi interni e l'interazione e la collaborazione tra individui o società. L'avvento del digitale ha cambiato la funzione della tecnologia, da elemento di supporto ai processi di business e ai modelli organizzativi esistenti a canale fondamentale attraverso cui si articolano i flussi di lavoro, in un rapporto di evoluzione simbiotica. A livello tecnologico, la trasformazione delle infrastrutture informatiche intrapresa nel corso degli ultimi anni vede l'affermarsi dell'adozione del paradigma Cloud e della transizione di un numero sempre più rilevante di servizi, con significativi benefici in termini di efficienza, scalabilità e portabilità. Oltre ai nuovi rischi associati a una più elevata complessità, le aziende dovranno confrontarsi con attacchi informatici sempre più sofisticati che spesso rendono le tradizionali misure di protezione insufficienti, soprattutto considerando i danni che oggi un attacco andato a buon fine può arrecare sia a livello economico sia di reputazione. Il cybercrime è diventato un vero e proprio business in cui gli attacchi continuano a essere sferrati attraverso tecniche evolute e riescono a truffare importanti aziende di tutti i settori con impatti economici notevoli. [3]

1.2 Lo sviluppo del rischio

Con l'aumentare dell'interconnessione nel mondo digitale, acquista un peso sempre maggiore il tema della sicurezza. Su Internet ci sono sempre nuovi pericoli che rappresentano una minaccia per le aziende. Il tema della sicurezza informatica è perciò attuale come mai prima d'ora, ma non riguarda solo la sicurezza su Internet, bensì si occupa anche di tutti gli altri aspetti e settori dell'Information Technology. La vastità del tema e il grado attuale di pericolo, specialmente provocato dall'aumento delle possibilità che il mondo digitale offre giorno dopo giorno, mostrano già come la sicurezza informatica sia molto importante. Basti pensare a quanti dati vengono

elaborati giornalmente sul computer, sul tablet o sullo smartphone, a quanti account vengono utilizzati su Internet per le diverse applicazioni, sulle tante piattaforme e a quanti dati bancari e altre informazioni sensibili vengono richieste nella quotidianità. Tutti questi dati sono intercettabili e alla mercé dei cyber criminali che li sfruttano per i loro scopi illeciti. Molti degli attacchi sono principalmente causati da presenza di malware o da una programmazione poco sicura e spesso facile da bypassare.

1.3 Situazioni di rischio attuali

Dal rapporto CLUSIT 2022 si evince che in 11 anni sono stati identificati, classificati e valutati oltre 15.000 attacchi informatici gravi. Di questi oltre la metà si sono verificati negli ultimi 4 anni e mezzo, a causa di un'accelerazione impressionante delle minacce cibernetiche. Lo scopo principale della ricerca CLUSIT è di elevare la consapevolezza e migliorare la comprensione del pubblico italiano rispetto all'evoluzione delle minacce cibernetiche, nell'ipotesi che il problema sarebbe inevitabilmente degenerato con grande rapidità. L'obiettivo è quello di offrire interessanti spunti di riflessione a coloro che si occupano di threat modeling, di cyber risk management e di cyber strategy, sia a livello aziendale che istituzionale, grazie ad una migliore "fotografia" dei rischi attuali resa possibile da questo ulteriore elemento di valutazione qualitativa delle dinamiche in atto. Osservando la situazione dal punto di vista quantitativo, confrontando i numeri del primo semestre 2018 con quelli del 2022 la crescita degli attacchi è stata del 53% (da 745 a 1.141). In 4 anni e mezzo la media mensile di attacchi gravi a livello globale è passata da 124 a 190. Oltre alla maggiore frequenza, la valutazione della Severity media di questi attacchi (indice di gravità degli attacchi analizzati) è drasticamente peggiorata, agendo da significativo moltiplicatore dei danni. L'osservazione di queste dinamiche conferma che a partire da 4 anni fa sia avvenuto un vero e proprio cambiamento epocale nei livelli globali di insicurezza informatica, ai quali evidentemente non è corrisposto un incremento sufficiente delle contromisure adottate dai difensori.

1.4 Analisi dei principali cyber attacchi a livello globale 2018-2022

Il campione che si esamina nelle pagine seguenti comprende 8.285 attacchi, classificati tra il gennaio 2018 e il giugno 2022 (oltre la metà del totale su 11 anni), di cui 1.874 nel 2020, 2.049 nel 2021 e 1.141 nel 1H 2022, con una media complessiva di 153 attacchi al mese nell'intero periodo considerato (erano 39 nel 2011, 130 nel 2018, 171 nel 2021 e sono 190 nel primo semestre 2022). Il picco massimo di sempre si è avuto a marzo 2022 (225 attacchi).

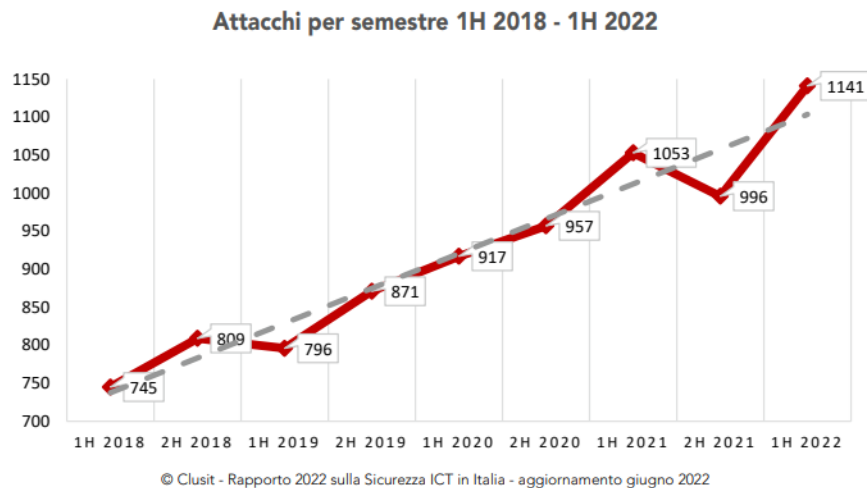


Figura 1.1: Attacchi per semestre

Si può anche osservare la distribuzione mensile degli attacchi registrati nel primo semestre del 2022.

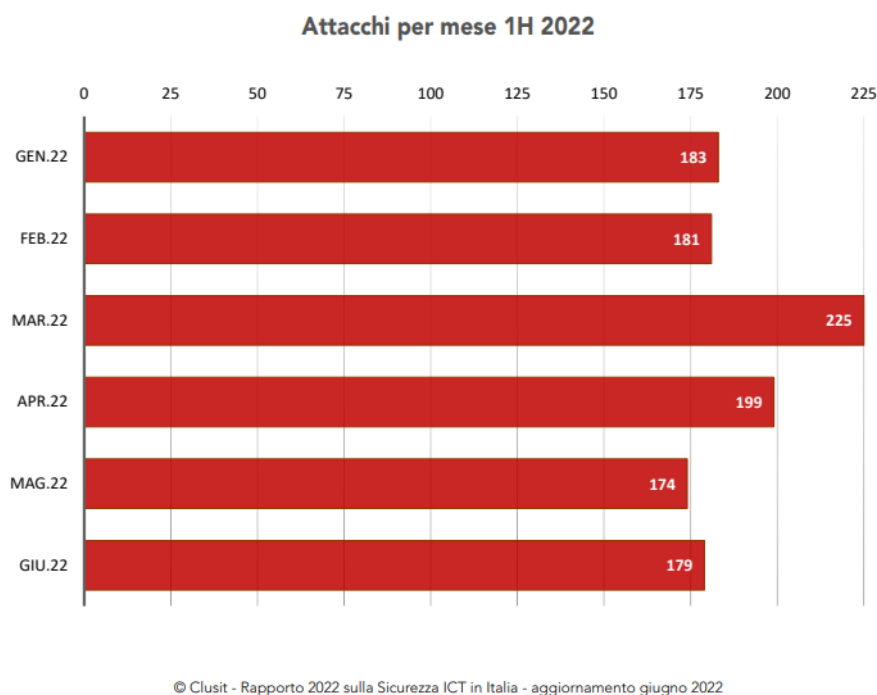


Figura 1.2: Attacchi mensili

Da questi dati si può iniziare ad evincere la gravità della situazione, in continuo peggioramento.

Le analisi e i relativi commenti si riferiscono ad un campione necessariamente parziale, per quanto ormai corposo e statisticamente significativo, rispetto al numero degli attacchi gravi effettivamente avvenuti nel periodo in esame. Questo accade sia perché un buon numero di aggressioni non diventano mai di dominio pubblico, oppure lo diventano ad anni di distanza (solitamente quanto più gli attacchi sono sofisticati), sia perché in molti casi è interesse delle vittime non pubblicizzare gli attacchi subiti, se non costretti dalle circostanze o da obblighi normativi particolari.

Capitolo 2

Minacce web note e risoluzioni

2.1 Il concetto di malware

Malware o “software malevolo” è un termine generico che descrive un programma/codice dannoso che mette a rischio un sistema. Ostili, invasivi e volutamente maligni, i malware cercano di invadere, danneggiare o disattivare computer, sistemi, reti, tablet e dispositivi mobili, spesso assumendo il controllo parziale delle operazioni del dispositivo. Lo scopo dei malware è riscuotere illecitamente a spese degli utenti. Sebbene i malware possono rubare, criptare o eliminare i dati, alterare o compromettere le funzioni fondamentali di un computer e spiare le attività degli utenti senza che questi se ne accorgano o forniscono alcuna autorizzazione, sono quindi anch’essi considerati veri e propri rischi. Per questa ragione è importante studiare e classificare i software malevoli, creando delle librerie di malware chiamate anche collezioni. Queste collezioni sono importantissime per il ricercatore che le utilizza per fare dei test su strumenti di sicurezza che dovrebbero riconoscerli, solitamente usati per testare antivirus.

Ci sono varie tipologie di malware in cui si potrebbe imbattersi:

- Gli adware sono software indesiderati progettati per presentare messaggi pubblicitari sullo schermo, spesso all’interno di un browser web. Generalmente, gli adware si avvalgono di un metodo subdolo, mascherandosi da componenti legittimi o nascondendosi in un altro programma al fine di provocarne con l’inganno l’installazione su PC, tablet o dispositivo mobile.
- Gli spyware sono malware che osservano segretamente le attività dell’utente sul computer, senza autorizzazione, per poi segnalarle al creatore del software.
- I virus sono malware che si attaccano ad altri programmi e, quando eseguiti, di solito inavvertitamente, si riproducono modificando altri programmi e infettandoli con il proprio codice.
- I worm sono malware simili ai virus, che si riproducono per diffondersi sugli altri computer di una rete e danneggiandoli, di solito, mediante la distruzione di dati e file.
- Un trojan o "cavallo di Troia" è uno dei malware più pericolosi. Di solito si presenta sotto forma di qualcosa di utile, per ingannare l’utente. Una volta entrato nel sistema, i criminali ottengono l’accesso non autorizzato al computer

della vittima. Da qui, i trojan possono essere utilizzati per rubare dati finanziari o installare altre minacce, come virus e ransomware.

- I ransomware sono malware che impediscono all'utente di accedere al proprio dispositivo e/o criptano i suoi file, obbligandolo a pagare un riscatto per riottenersi. I ransomware sono stati definiti "l'arma scelta" dei criminali, perché richiedono un pagamento rapido e ingente in criptovalute difficili da rintracciare. Il codice alla base dei ransomware è semplice da ottenere sui marketplace criminali e difendersi da essi è molto difficile.
- I rootkit sono dei malware che forniscono al criminale i privilegi da amministratore del sistema infetto. Generalmente, sono progettati per rimanere nascosti agli occhi dell'utente, degli altri software e del sistema operativo stesso.
- I keylogger sono malware che registrano la pressione dei tasti degli utenti sulla tastiera, memorizzando le informazioni raccolte e inviandole ai criminali responsabili, che puntano a informazioni sensibili come nomi utente, password o dati delle carte di credito.
- Il cryptomining dannoso, noto anche come drive-by mining o cryptojacking, è una tecnica malware sempre più diffusa, che in genere prevede l'installazione da parte di un trojan. Consente a persone estranee di utilizzare un computer per "generare" (mining) criptovalute, ad es. Bitcoin o Monero. Anziché lasciare che gli utenti raccolgano i frutti del lavoro del proprio computer, i cryptominer inviano la valuta raccolta ai propri account. Essenzialmente, un cryptominer dannoso deruba gli utenti delle loro risorse per lucro.

2.1.1 Come proteggersi dai malware

Presentiamo in seguito delle best practice da seguire per prevenire un attacco malware:

- Prestare particolare attenzione ai domini che finiscono con strane combinazioni di lettere, ossia in qualcosa di diverso da com, org, edu o biz, per dirne alcuni, poiché possono indicare la pericolosità di un sito web.
- Non fare clic su pop-up pubblicitari mentre si naviga su internet. Non aprire allegati e-mail non richiesti o scaricare software da siti web affidabili.
- Assicurarsi che il sistema operativo, i browser e i plugin che utilizzi siano sempre aggiornati.
- Non scaricare app da terze parti.
- Non fare clic sui link non verificati contenuti in e-mail, SMS e messaggi di origine sconosciuta.
- Fare uso di un buon programma anti-malware.

2.1.2 Attacchi Malware recenti

Uno degli attacchi ransomware più recenti è avvenuto all'inizio di gennaio 2023 a Yum! Questo attacco ha causato la chiusura di 300 punti vendita dei suoi marchi KFC, Pizza Hut e Taco Bell. Questo attacco ransomware sembra essere stata una doppia estorsione che ha messo a rischio il furto di database, dati riservati e sistemi dell'operatore del marchio. [9]

2.2 Attacchi DDoS

Un attacco DDoS (Distributed Denial of Service) è un tentativo ostile di bloccare il normale traffico di un server, servizio o rete sopraffacendo la vittima o l'infrastruttura circostante inondandola di traffico Internet. La conseguenza più comune di un attacco DDoS è che un sito o un servizio diventi improvvisamente lento o non disponibile. [4]

2.2.1 Attacchi DDoS recenti

Ad oggi, il più grande attacco DDoS è avvenuto nel settembre del 2017. L'attacco era diretto ai servizi di Google e ha raggiunto una dimensione di 2,54 Tbps. Gli aggressori hanno inviato pacchetti spoofed a 180.000 server Web, che a loro volta hanno inviato risposte a Google. L'attacco non è stato un incidente isolato: negli ultimi sei mesi gli aggressori avevano diretto più attacchi DDoS all'infrastruttura di Google.

2.3 Attacchi zero-day

Gli attacchi zero-day, noti anche come exploit zero-day, sono tentativi riusciti da parte dei criminali informatici di trovare e sfruttare vulnerabilità del software fino a quel momento sconosciute. Sfortunatamente, in tutto il software sono presenti punti deboli che offrono backdoor sfruttabili dagli hacker per inserire malware o violare i dati. Gli attacchi che sfruttano vulnerabilità precedentemente sconosciute dagli ingegneri software sono chiamati attacchi "zero day" proprio perché gli sviluppatori hanno avuto a disposizione zero giorni per risolvere il problema prima dell'attacco. Gli attacchi zero-day sono violazioni della sicurezza particolarmente pericolose proprio perché la loro estensione è sconosciuta. In risposta a un attacco, uno sviluppatore di software può creare una patch, che però non è di alcun aiuto per gli utenti già colpiti.

A peggiorare le cose, molti sistemi antivirus tradizionali utilizzano strumenti di rilevamento delle minacce che si basano sul trovare una corrispondenza tra firme rivelatrici e attacchi informatici noti. Poiché gli attacchi zero-day sfruttano vulnerabilità precedentemente sconosciute (e forse malware non ancora noti), non solo possono rimanere inosservati per lunghi periodi di tempo, ma è anche molto più difficile difendersi. [1]

2.3.1 Attacchi zero-day recenti

Uno degli attacchi Zero-day più recenti è stato ritrovato nel sistema operativo CISCO IOS XE. Si tratta di Privilege Escalation ed è stimato un attacco grave dal momento che se sfruttata la vulnerabilità da parte di un utente malintenzionato non autenticato, permette ad esso la creazione di un account di tipo admin dall'interfaccia Web UI e riuscirebbe ad ottenere il controllo dei dispositivi target. [5]

2.4 OWASP : Vulnerabilità comuni e classificazioni

OWASP è un'iniziativa open-source internazionale no-profit nata nel 2001 con lo scopo di realizzare una serie di linee guida ma anche di strumenti pratici, utili a supportare i web developer nello sviluppo sicuro di web application. Trattandosi di un progetto open source, tutte le risorse messe a disposizione della community sono liberamente consultabili da chiunque voglia mettere la sicurezza al centro del suo progetto di sviluppo.

Le tecniche e i principi definiti da OWASP stanno diventando delle best practices riconosciute a livello internazionale per la prevenzione delle vulnerabilità informatiche.

Tramite l'applicazione sistematica di queste procedure lo sviluppatore può ridurre considerevolmente il rischio di data breach causato da attacchi hacker e agevolato da una scarsa qualità del codice. Si tratta per il momento di indicazioni facoltative, ma in diversi contesti si sta cercando di renderle obbligatorie per lo sviluppo web.

Un passo in avanti in questo senso si è avuto con l'adozione del GDPR (Regolamento generale per la protezione dei dati personali) che richiede il rispetto della protezione dei dati fin dalla progettazione di un software, secondo il principio di privacy by design.

2.4.1 Ultimi cambiamenti Owasp Top 10

Ci sono tre nuove categorie nella Owasp Top 10 per il 2021, si possono qui vedere i cambiamenti e le differenze tra la Owasp Top 10 stilata nel 2017 e la Owasp Top 10 del 2021. Subito sotto verranno riportate le descrizioni di ogni categoria illustrata della sola versione più recente.

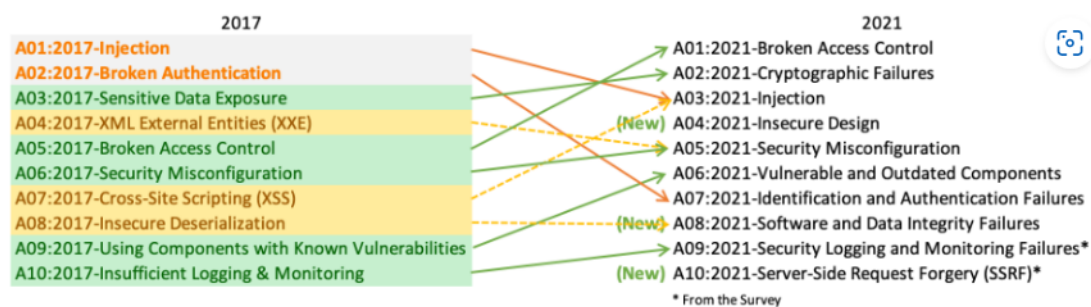


Figura 2.1: Cambiamenti top10

2.4.2 Broken Access Control

Il controllo degli accessi fa rispettare la policy in modo che gli utenti non possano agire al di fuori dei permessi previsti. Problematiche su questo tipo di controllo tipicamente portano alla divulgazione non autorizzata di informazioni, alla modifica o alla distruzione di tutti i dati o l'esecuzione di una funzione di business al di fuori dei limiti dell'utente.

2.4.3 Prevenire Broken Access Control

Il controllo degli accessi è efficace solo nel codice lato server o API server-less, dove l'attaccante non può modificare i meccanismi di controllo dell'accesso o i metadati.

- Tranne che per le risorse pubbliche, applicare il principio di deny by default.
- Implementare i meccanismi di controllo dell'accesso una volta sola e utilizzarli in tutta l'applicazione, incluso limitare l'utilizzo di Cross-Origin Resource Sharing (CORS).
- I controlli di accesso del Model dovrebbero imporre la proprietà dei record piuttosto che accettare che l'utente possa creare, leggere, aggiornare o cancellare qualsiasi record.
- I requisiti unici dei vincoli di business di un'applicazione dovrebbero essere applicati nei modelli di dominio.
- Disabilitare il directory listing del server web e garantire che i metadati dei file (ad es, .git) e i file di backup non siano presenti all'interno delle web roots.
- Registrare i fallimenti dei meccanismi di controllo dell'accesso, avvisare gli amministratori quando appropriato (ad es, fallimenti ripetuti).
- Implementare meccanismi di rate limiting per accesso all'API e al controller per minimizzare il danno da strumenti di attacco automatizzati.
- Gli identificatori di sessione stateful dovrebbero essere invalidati sul server dopo il logout. I token JWT stateless dovrebbero piuttosto essere di breve durata in modo che la finestra di opportunità per un attaccante sia ridotta al minimo. Per i JWT di lunga durata è altamente raccomandato di seguire gli standard OAuth per revocare l'accesso.

2.4.4 Cryptographic Failures

Il Cryptographic Failures comprende tutti quegli errori legati alla crittografia che spesso portano all'esposizione di dati sensibili o alla compromissione del sistema sia quanto memorizziamo sia quando inviamo informazioni sensibili. Il primo passo è determinare le esigenze di protezione dei dati. Per esempio, password, numeri di carte di credito, documenti sanitari, informazioni personali e segreti aziendali richiedono una protezione adeguata, soprattutto se quei dati ricadono sotto le leggi sulla privacy, ad es. General Data Protection Regulation (GDPR), o regolamenti, ad es, protezione dei dati finanziari come il PCI Data Security Standard (PCI DSS).

Per tutte queste tipologie di dati bisogna porre maggiore attenzione nel momento in cui: i dati sono trasmessi in chiaro, sono implementati algoritmi o protocolli crittografici vecchi o deboli, sono utilizzate chiavi crittografiche deboli, vecchie e/o riutilizzate, non viene applicata adeguata crittografia, le password sono chiavi crittografiche, vengono utilizzate funzioni hash deprecate.

2.4.5 Prevenire Cryptographic Failures

- Identificare quali dati sono sensibili secondo le leggi sulla privacy, requisiti normativi o esigenze aziendali.
- Non conservare inutilmente i dati sensibili. Eliminarli il prima possibile.
- Assicurarsi di cifrare tutti i dati sensibili a riposo.
- Utilizzare algoritmi, protocolli e chiavi standard forti e aggiornati. Avere un adeguato processo di key management.
- Crittografare tutti i dati in transito con protocolli sicuri come TLS con cifrari che garantiscano la FS (forward secrecy).
- Disabilitare il caching per le risposte che contengono dati sensibili.
- Usare sempre un meccanismo di crittografia autenticata invece della semplice crittografia.
- Evitare funzioni crittografiche e schemi di padding deprecati
- Assicuratevi che la randomness crittografica venga utilizzata laddove appropriato, e che non sia stato utilizzato un seed prevedibile o con bassa entropia.

2.4.6 Injection

Una applicazione è vulnerabile alle injection quando i dati forniti dall'utente non sono validati, filtrati o sanificati, le query dinamiche senza escaping contestuale vengono passate direttamente all'interprete, input malevolo viene utilizzato in modo diretto o concatenato.

Le forme più comuni di injection che si possono trovare sono: SQL, NoSQL, OS command (quando il payload inserito dall'utente malintenzionato viene eseguito come comando del sistema operativo), Expression Language (quando l'attaccante controlla i dati in entrata dell'interprete). Questi risultano solo alcuni attacchi injection ma ce ne possono essere di molte altre tipologie.

2.4.7 Prevenire Injection

- Per qualsiasi query dinamica è consigliabile svolgere l'escape dei caratteri speciali usando la sintassi di escape specifica per quell'interprete.
- Usare controlli SQL all'interno delle query per prevenire la divulgazione di massa dei record, questo in caso di SQL injection.
- Utilizzare validazione degli input lato server.

- Per qualsiasi query dinamica è consigliabile svolgere l'escape dei caratteri speciali usando la sintassi di escape specifica per quell'interprete.
- Usare controlli SQL all'interno delle query per prevenire la divulgazione di massa dei record, questo in caso di SQL injection.

2.4.8 Insecure Design

Insecure Design viene definita come una macro categoria che include tutte le problematiche legate alla progettazione inefficace o mancante dei controlli di sicurezza in un'infrastruttura web-based. Negli ultimi decenni, quest'area d'indagine ha visto grandi progressi grazie soprattutto al continuo incremento degli attacchi informatici a danno delle infrastrutture web-based e cloud-based.

Quando si parla di design insicuro di un'applicazione web si fa riferimento ad una infrastruttura costruita su basi non solide: le vulnerabilità si trovano nel codice sorgente. Al contrario quando si parla di difetti di implementazione software parliamo di una struttura di base sicura ma che viene aggiornata con pezzi di codice che comportano la comparsa di vulnerabilità a lungo termine.

La progettazione insicura di un software o di implementazione in ogni caso, comportano una serie di pericoli che sono provocati dal trascurare gli standard di sicurezza informatica minimi. Ad ogni modo, la presenza di vulnerabilità di InSecure Design costituisce un problema non da poco considerato che le minacce informatiche sono in continua evoluzione e risolvere queste problematiche significa riprogettare logiche e funzionamenti del software.

2.4.9 Prevenire Insecure Design

- Stabilire e utilizzare un ciclo di vita di sviluppo sicuro con i professionisti di AppSec per aiutare a valutare e progettare la sicurezza e i controlli relativi alla privacy.
- Stabilire e utilizzare una libreria di design pattern sicuri o componenti pronti all'uso.
- Integrare i controlli di plausibilità ad ogni livello della vostra applicazione (dal frontend al backend).
- Scrivere test unitari e di integrazione per convalidare che tutti i flussi critici siano resistenti al modello di minaccia rappresentato. Compilare i casi d'uso e i casi di uso improprio per ogni livello della vostra applicazione.
- Limitare il consumo di risorse per utente o servizio.

2.4.10 Security Misconfiguration

Una applicazione potrebbe risultare vulnerabile se ci sono permessi configurati in maniera impropria sui servizi cloud, se sono abilitate o installate funzioni non necessarie, se account di default sono ancora abilitati con password predefinite, a seguito di condizione di errore vengono rivelati agli utenti messaggi di errore verbosi, se nei sistemi aggiornati le funzionalità di sicurezza non vengono abilitate o configurate

nella maniera opportuna, se le impostazioni di sicurezza non sono configurati in maniera consona su valori sicuri e/o se il software non è aggiornato.

2.4.11 Prevenire Security Misconfiguration

- Un processo di hardening (insieme di operazioni specifiche di configurazione di un dato sistema informatico che mira a minimizzare l'impatto di possibili attacchi informatici) ripetibile rende veloce e facile il deploy (è il processo di installazione, configurazione, aggiornamento ed abilitazione di un'applicazione o di una suite di applicazioni che consente di utilizzare un sistema software) di un altro ambiente preconfigurato in modo sicuro. Gli ambienti di sviluppo dovrebbero essere tutti configurati in modo speculare, con credenziali diverse per ogni ambiente. Questo processo dovrebbe essere automatizzato per minimizzare lo sforzo richiesto per impostare un nuovo ambiente configurato in modo sicuro.
- Rimuovere o non installare funzionalità e framework inutilizzati.
- Inserire un task per rivedere ed aggiornare le configurazioni appropriate a tutte le security notes.
- L'invio di direttive di sicurezza ai client, ad esempio, i Security Headers.
- Un processo automatizzato per verificare l'efficacia delle configurazioni e impostazioni in tutti gli ambienti.

2.4.12 Vulnerable and Outdated Components

La vulnerabilità si presenta nel caso in cui il software risulta vulnerabile, non supportato o non aggiornato, se non si fa la scansione delle vulnerabilità di libreria regolarmente e non si consultano i report di sicurezza relativi ai componenti utilizzati, se non si corregge/aggiorna la piattaforma sottostante, i framework, le dipendenze in maniera tempestiva, se gli sviluppatori non testano la compatibilità delle librerie aggiornate o delle librerie nuove o se i componenti non vengono configurati in modo sicuro.

2.4.13 Prevenire Vulnerable and Outdated Components

Per prevenire questa tipologia di vulnerabilità bisognerebbe:

- Rimuovere le dipendenze, le funzionalità, i componenti, i file, e la documentazione non utilizzate.
- Inventariare in modo continuo le versioni dei componenti lato client e lato server (ad esempio, framework, librerie) e le loro dipendenze usando strumenti come OWASP.
- Monitorare continuamente fonti come Common Vulnerability and Exposures (CVE) e il National Vulnerability Database (NVD) per vulnerabilità nei componenti.

- Ottenere i componenti solo da fonti ufficiali tramite link sicuri. Preferire pacchetti firmati per ridurre la possibilità di includere un componente modificato o dannoso.
- Controllare le librerie e i componenti che non sono più mantenuti o che non sviluppano più patch di sicurezza per le vecchie versioni. Se il patching non è possibile, considerare l'implementazione di una patch virtuale per monitorare, rilevare o proteggere dal problema individuato.

2.4.14 Identification and Authentication Failures

Precedentemente sotto il nome di Broken Authentication ci possono essere debolezze sui meccanismi di autenticazione nel caso in cui l'applicazione permette attacchi automatici dove l'attaccante ha una lista di coppie nome utente e password validi, se permette attacchi di brute force o altri attacchi automatici, se permette password di default deboli, se presenta un recupero delle credenziali e delle password deboli, se memorizza le password in chiaro, se non ha un sistema di autenticazione a due fattori, se riutilizza l'identificatore di sessione dopo un login avvenuto con successo, non l'indicatore di sessione.

2.4.15 Prevenire Identification and Authentication Failures

Per prevenire questa tipologia di vulnerabilità bisognerebbe:

- Implementare l'autenticazione a più fattori per prevenire attacchi di credential stuffing, brute force e riutilizzo delle credenziali rubate.
- Non mettere in produzione sistemi con credenziali di default, in particolare per gli utenti admin.
- Implementare controlli sulla debolezza delle password, come verificare le password nuove o modificate con la lista delle 10,000 password peggiori.
- Allineare i requisiti di lunghezza delle password, complessità e politiche di rotazione con le linee guida della sezione 5.1.1 del documento 800-63b pubblicato dal National Institute of Standards and Technology (NIST) riguardante la memorizzazione dei secret o altre policy relative alle password moderne.
- Limitare o ritardare sempre più i tentativi di login falliti, ma fare attenzione a non creare uno scenario di denial of service. Loggare tutti i tentativi falliti e avvertire gli amministratori quando vengono rilevati attacchi di credential stuffing, brute force o altri.

Autenticazione a due fattori

Al giorno d'oggi non risulta più sufficiente proteggere i propri account con password forti, è consigliabile utilizzare una autenticazione forte conosciuta come autenticazione a due fattori.

Risulta ormai noto l'importanza di utilizzare password molto complesse e diverse, come allo stesso tempo utilizzare strumenti per riuscire a ricordare tutte le password

che ci ritroviamo a gestire. Per risolvere questa problematica si utilizzano dei password manager, applicazioni che hanno la funzione di mantenere le nostre password in maniera crittografata.

Tutto questo però non basta a garantire la sicurezza di una password, in generale una autenticazione basata sull'utilizzo di una sola password risulta debole.

Per questa motivazione è stata introdotta l'autenticazione a due fattori che rappresenta uno dei sistemi per mantenere al sicuro i nostri dati.

Solitamente l'autenticazione può avvenire in diversi modi:

- Conoscenza: una cosa che sai (password o PIN)
- Possesso: una cosa che hai (smartphone o token di sicurezza)
- Inerenza: una cosa che sei (impronte digitale, timbro vocale etc..)

L'autenticazione a due fattori avviene nel momento in cui si utilizzano due dei tre fattori sopra elencati.

Solitamente nell'autenticazione a due fattori in seguito all'inserimento della password verrà richiesto un secondo fattore.

Risulta indispensabile che il secondo fattore sia inattaccabile, in caso contrario il problema dell'effettuare una autenticazione sicura non sarebbe risolto.

Grazie all'affinarsi delle tecniche di hacking mediante Intelligenza artificiale e il deepfake che consente di imitare volti e voci, la cyber security in senso ampio viene messa in difficoltà, è dunque sconsigliato l'utilizzo di Inerenza.

Da linee guida del NIST si consiglia quindi di ottenere il secondo fattore attraverso lo smartphone (sotto forma di sms o tramite un'apposita applicazione) o tramite un token fisico perché generato in maniera pseudocasuale secondo uno specifico algoritmo ed ha una durata molto limitata nel tempo (solitamente 30 secondi). Per questo motivo, lo si definisce anche OTP: "one time password".

Acronimo di "One-Time-Password", le OTP sono password che possono essere usate una sola volta e questo, già da solo, costituisce una garanzia in più rispetto alle password tradizionali, sempre identiche e utilizzabili per un periodo di tempo variabile a seconda delle policy aziendali. Le OTP sono una combinazione di caratteri numerici o alfanumerici più difficili da violare perché possono essere usate una sola volta e, di norma almeno, vengono inviate a un dispositivo in dotazione all'utente. Esistono due tipologie di OTP utilizzabili:

- TOTP, password monouso valide per un periodo limitato di tempo solitamente per 15-60 secondi. Una volta scaduto il tempo assegnato, il codice non è più utilizzabile ed è necessaria l'assegnazione di uno nuovo. Più la vita del codice è breve, più è difficile violarlo perché una persona non autorizzata deve ottenere il codice in un lasso di tempo ristretto.
- HOTP, password monouso basate su HMAC, ossia un codice di autenticazione che fa leva su una funzione di hash. Mediante HMAC sono garantite l'integrità e l'autenticità di un messaggio, è una metodologia quasi non più utilizzata.

2.4.16 Software and Data Integrity Failures

Le problematiche dell'integrità del software e dei dati riguardano il codice e l'infrastruttura che non ne verificano adeguatamente l'integrità. Un esempio è quando un'applicazione si affida a plugin, librerie o moduli da fonti, repository e content delivery network (CDN) non attendibili.

2.4.17 Prevenire Software and Data Integrity Failures

- Usare firme digitali o meccanismi equivalenti per verificare che il software o i dati provengano dalla fonte prevista e non siano stati alterati.
- Assicuratevi che venga usato uno strumento di sicurezza della supply chain del software, come OWASP Dependency Check o OWASP CycloneDX, per verificare che i componenti non contengano vulnerabilità note.
- Assicurarsi che ci sia un processo di revisione per le modifiche al codice e alla configurazione per ridurre al minimo la possibilità che codice o configurazione dannosi vengano introdotti nella pipeline del software.
- Assicurarsi che la pipeline CI/CD sia adeguatamente segregata, configurata adeguatamente e sia presente un meccanismo di controllo degli accessi per assicurare l'integrità del codice che passa attraverso i processi di compilazione e distribuzione.
- Assicuratevi che i dati serializzati non firmati o non crittografati non vengano inviati ai client non fidati senza qualche forma di controllo dell'integrità o firma digitale per rilevare la manomissione o il replay dei dati serializzati.

2.4.18 Security Logging and Monitoring Failures

Questa categoria è per aiutare a rilevare, svolgere escalation e rispondere alle violazioni attive. Senza logging e monitoraggio, le violazioni non possono essere rilevate. Il logging, il rilevamento, il monitoraggio e la risposta attiva insufficienti si verificano ogni volta che gli eventi verificabili come il login, i login falliti e le transazioni ad alto valore non vengono registrate, se warning ed errori sono inadeguati/poco chiari, se i file di log vengono memorizzati solo localmente, se non sono presenti le soglie di allarme/processi di escalation della risposta, i penetration test e le scansioni non danno nessun allarme, se l'applicazione non è in grado di rilevare, svolgere escalation o avvisare per gli attacchi attivi in real-time o quasi real time. Si è poi inoltre vulnerabili alla fuga di informazioni se gli eventi di logging e gli alert sono visibili ad un utente o ad un attaccante.

2.4.19 Prevenire Security Logging and Monitoring Failure

- Assicurarsi che tutti i login, il controllo degli accessi e gli errori a seguito della verifica degli input lato server possono essere registrati con un contesto utente sufficiente per identificare account sospetti o malevoli e conservati per un tempo sufficiente a consentire un'analisi forense a posteriori.

- Assicurarsi che i dati di log siano codificati correttamente per prevenire injection o attacchi ai sistemi di registrazione o monitoraggio.
- Assicurarsi che le transazioni di alto valore abbiano un audit trail con controlli di integrità per prevenire manomissioni o cancellazioni, come le tabelle append-only di un database, o simili.
- I team DevSecOps dovrebbero stabilire sistemi di monitoraggio di allerta efficaci in modo che le attività sospette siano rilevate e affrontate rapidamente.
- Stabilire o adottare un piano incident response e recovery, come ad esempio il National Institute of Standards and Technology (NIST) 800-61r2 o successivo.

2.4.20 Server-Side Request Forgery

Le falle SSRF si verificano ogni volta che un'applicazione web recupera una risorsa remota senza validare l'URL fornito dall'utente. Questo permette ad un attaccante di forzare l'applicazione ad inviare una richiesta preparata ad hoc ad una destinazione inattesa, anche quando è protetta da un firewall, una VPN o un altro tipo di network access control list (ACL).

Dato che le moderne applicazioni web forniscono agli utenti finali parecchie funzionalità, scaricare dati da un URL è uno scenario comune. Di conseguenza, l'incidenza di SSRF è in crescita. Inoltre, la gravità di SSRF sta diventando più alta a causa dei servizi cloud e alla complessità crescente delle architetture.

2.4.21 Prevenire Server-Side Request Forgery

Dal layer di rete:

- Segmentare in reti separate le funzionalità che richiedono un accesso alle risorse remote per ridurre l'impatto di SSRF.
- Applicare politiche di firewall "deny by default" o regole di controllo per bloccare tutto il traffico intranet tranne quello essenziale.

Dal layer applicativo:

- Sanitizzare e convalidare tutti i dati di input forniti dal cliente.
- Far rispettare lo URL schema, la porta e la destinazione con una allow list
- Non inviare risposte raw ai client.

Praticità

Non mitigare la SSRF attraverso l'uso di una deny list o di un'espressione regolare. Gli attaccanti hanno a disposizione liste di payload, strumenti e abilità per bypassare le deny list.

Capitolo 3

La valutazione del rischio digitale per applicazioni web

3.1 Risk assessment e risk management

La valutazione dei rischi digitali è fondamentale per le organizzazioni che intendono avere la piena consapevolezza della loro situazione e desiderino affrontare un percorso di miglioramento continuo. Formalmente, l'analisi del rischio (risk analysis) è parte della valutazione del rischio (risk assessment), a sua volta parte della gestione del rischio (risk management). Quest'ultima è organizzata in più fasi interconnesse:

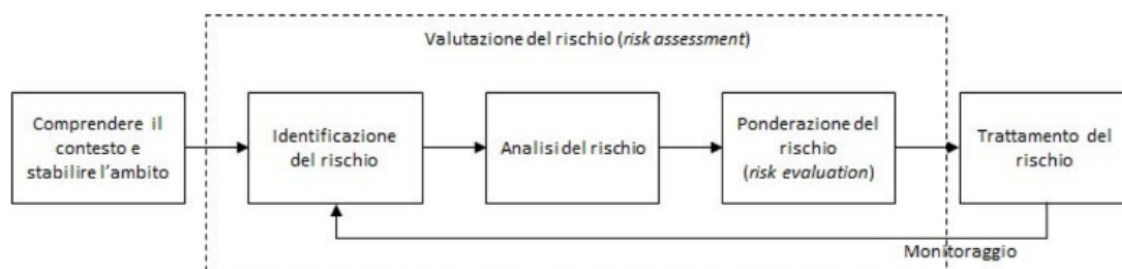


Figura 3.1: Valutazione del rischio

Alcuni testi indicano come “gestione del rischio” la sola fase di trattamento del rischio. L’approccio deve essere sistematico, ossia tale da rendere evidente, e quindi ripetibile, attraverso i dati, le elaborazioni e le valutazioni, il ragionamento che ha portato alle scelte fatte. Questo permette anche di verificare, validare e affinare nel tempo il proprio approccio alla gestione del rischio, riducendo conseguentemente l’aleatorietà delle valutazioni fatte.

I principali passi della gestione del rischio consistono in:

- Comprensione del contesto e stabilire l’ambito:
 - Analisi del contesto: consente di cogliere tutti gli elementi che contribuiscono allo sviluppo di una conoscenza del rischio e di adattare il processo di analisi del rischio alle effettive esigenze del settore di attività.
 - Identificazione dell’approccio da adottare e dei framework di riferimento: la scelta dell’approccio da adottare rende la gestione del rischio pertinente, inquadrando gli elementi più significativi e prendendo in considerazione

tutti i tipi di rischi applicabili alle attività e funzioni. L'adozione di un framework permette di stabilire un dizionario di termini e misure condivise con tutti.

- Mappatura dei processi, inclusa la filiera di fornitura: costituisce il punto di partenza per una descrizione dettagliata dell'attività dell'organizzazione e, quindi, per l'identificazione delle aree e attività più critiche.
 - Identificazione del perimetro da analizzare.
 - Assegnazione delle responsabilità: l'individuazione puntuale dei risk owner intesi come persone responsabili dell'analisi delle minacce e delle vulnerabilità in specifici ambiti e, conseguentemente, della gestione del rischio anche nelle attività quotidiane. Devono avere i necessari poteri per attuare le adeguate misure di mitigazione dei rischi e interagire con i vertici e le persone incaricate del coordinamento complessivo della gestione dei rischi.
- Valutazione del rischio: identificazione e caratterizzazione dei pericoli e dell'esposizione a essi, quantificazione del rischio, in relazione alle minacce esistenti e alla probabilità che queste possano concretizzarsi, e valutazione della sua accettabilità in relazione al contesto e alle strategie dell'organizzazione. Si divide in:
 - identificazione del rischio: Il metodo per identificare le vulnerabilità conosciute (ed elencate nel CVE) e sconosciute, di un sistema è il cosiddetto Vulnerability Assessment (VA). Possiamo suddividere VA in 4 fasi:
 - * La prima fase prevede un vulnerability scanning, eseguito con strumenti automatici di scansione delle vulnerabilità ; esso permette di individuare una buona parte delle vulnerabilità già note.
 - * La seconda fase prevede un security test evaluation (STE), con il quale si svolgono dei test di base tesi alla ricerca delle vulnerabilità più semplici.
 - * Successivamente si eseguono penetration test (PT) effettuati da persone con elevate competenze tecniche.
 - analisi del rischio: l'esecuzione di un'analisi dei rischi è un'attività ricca di incognite, sia per quanto concerne la sua conduzione, sia per quanto attiene i risultati ottenuti. È quindi particolarmente importante che si abbia consapevolezza di questi limiti: ci possono essere errori negli approcci utilizzati, la conoscenza imperfetta della minaccia, nuove vulnerabilità da scoprire (etc..) e per questa motivazione dovrebbero essere adeguatamente documentate le scelte e gli approcci effettuati.
 - ponderazione dei rischi: confronto del livello di rischio residuo con i criteri di accettazione condivisi con i vertici dell'organizzazione.
 - Trattamento dei rischi: valutazione delle alternative disponibili per affrontare il rischio (per esempio, mitigandolo o accettandolo).
 - Riesame del rischio (risk monitoring and review): attività in cui sono raccolte e analizzate informazioni relative ai rischi in modo da verificare l'efficacia del loro trattamento.

3.2 Integrazione continua DevSecOps

DevSecOps è una pratica di integrazione dei test di sicurezza in ogni fase del processo di sviluppo del software chiamato DevOps.

DevOps utilizza strumenti e automazione per promuovere una maggiore collaborazione, comunicazione e trasparenza tra team.

La Continuous Integration and Continuous Delivery (CI/CD) è una moderna pratica di sviluppo software che utilizza passaggi automatizzati di compilazione e test per fornire in modo affidabile ed efficiente piccole modifiche all'applicazione. Gli sviluppatori utilizzano gli strumenti CI/CD per rilasciare nuove versioni di un'applicazione.

Includere strumenti e processi che incoraggiano la collaborazione tra sviluppatori, specialisti della sicurezza e team operativi per creare software efficiente e sicuro. DevSecOps porta una trasformazione culturale che rende la sicurezza una responsabilità condivisa per tutti coloro che stanno costruendo il software. DevSecOps introduce la sicurezza nella pratica DevOps integrando le valutazioni della sicurezza in tutto il processo CI/CD. Rende la sicurezza una responsabilità condivisa tra tutti i membri del team coinvolti nella creazione del software. [2] [14]

3.2.1 Vantaggi DevSecOps

L'impiego della metodologia DevSecOps consente di rilasciare software migliore in maniera più veloce, identificando nativamente le possibili vulnerabilità che potrebbero dare luogo a seri problemi in seguito ai deploy pianificati lungo la roadmap dell'applicazione.

3.2.2 Pipeline DevSecOps

Le pipeline DevSecOps, conosciute anche come pipeline CI/CD, consistono in una successione di script eseguiti in sequenza per la realizzazione di una nuova versione del software. L'introduzione di monitoraggio e automazione mira a potenziare lo sviluppo, il testing e il rilascio dell'applicativo. Pur essendo possibile eseguire ogni passaggio manualmente, l'integrazione all'interno di una pipeline garantisce un processo più rapido ed efficiente.

I passaggi che costituiscono una pipeline CI/CD sono una serie di attività raggruppate in ciò che è noto come fase della pipeline. Le fasi tipiche della pipeline includono:

- Build: si compila l'applicazione e si testano le funzionalità.
- Test: in questa fase si eseguono i test di sicurezza.
- Release: in questa fase il codice viene rilasciato in ambiente di testing.

Se vengono eseguite con successo le precedenti fasi citate, avverrà il deploy in produzione. Ecco un esempio in cui si effettua il deploy, successivamente potremmo osservare il fallimento di alcuni di questi campi.

Stage View

	Declarative: Checkout SCM	Git Clone	App Release	Build Application	SonarQube analysis	OWASP Dependency-Check execution	Store artifacts	Deploy configuration	Deploy db	WildFly - Stop servers	WildFly - Undeploy	WildFly - Deploy	WildFly - Start servers
Average stage times: (Average full run time: ~6min 50s)	1s	24s	6s	2min 29s	120ms	128ms	374ms	22s	125ms	3min 8s	35s	32s	24s
#1718 Oct 03 13:19 5 commits	1s	26s	6s	2min 40s	123ms	148ms	490ms	22s	143ms	4min 53s	37s	32s	3s
#1717 Oct 02 16:09 3 commits	1s	25s	6s	2min 26s	124ms	136ms	400ms	20s	118ms	3min 18s	36s	32s	3s
#1716 Oct 02 17:09 8 commits	1s	26s	6s	2min 27s	101ms	113ms	357ms	23s	148ms	14s	34s	31s	4s
#1715 Oct 02 16:38 2 commits	1s	25s	7s	2min 27s	125ms	128ms	382ms	21s	116ms	3min 19s	35s	32s	3s

Figura 3.2: Pipeline di successo

In questa immagine riportata sopra possiamo osservare queste diverse fasi testate dai plugin scaricati dall'applicazione utilizzata che verrà analizzata in seguito.

3.3 Analisi dei report

Alcuni strumenti che esamineremo offriranno agli utenti la possibilità di generare report al fine di segnalare ogni tipo di possibile pericolo individuato.

Il processo coinvolge scansioni automatizzate e la valutazione delle vulnerabilità presenti in un sistema, con l'obiettivo di individuare e correggere potenziali punti deboli che potrebbero essere sfruttati da un attaccante.

L'inizio del processo consiste nelle scansioni automatizzate del sistema, utilizzando strumenti software specializzati noti come Vulnerability Scanner. Questi scanner esaminano il sistema alla ricerca di vulnerabilità, come falle di sicurezza nel sistema operativo, applicazioni web, database, firewall e altri componenti del sistema.

Dopo il completamento della scansione, il Vulnerability Scanner prosegue con la valutazione delle vulnerabilità individuate per determinarne la gravità e il rischio associato. Tale valutazione si basa su metriche standard come il Common Vulnerability Scoring System (CVSS), che classifica le vulnerabilità in base alla loro gravità e alla loro rilevanza per l'organizzazione.

Per ogni vulnerabilità individuata, è disponibile una dettagliata descrizione, il livello di rischio, la localizzazione precisa della vulnerabilità e le indicazioni per risolvere il problema.

Il compito del pentester consiste nel verificare la validità delle segnalazioni. Se una segnalazione è valida, il problema deve essere corretto; in caso contrario, la vulnerabilità individuata può essere dichiarata erroneamente positiva.

3.4 Tecnologie DevSecOps

L'utilizzo di strumenti DevSecOps richiede un'automazione completa, limitando, in casi eccezionali, qualsiasi intervento manuale e evitando modifiche alle configurazioni e script personalizzati.

Gli strumenti DevSecOps devono garantire una risposta in tempo reale, poiché la velocità dei cicli CI/CD e la riduzione del time to market delle applicazioni rimangono tra i principali driver per l'adozione delle metodologie di sviluppo agili. Ottimizzare anche i requisiti di sicurezza dell'applicazione entro tempi ragionevoli costituisce un valore aggiunto fondamentale.

L'automazione e la velocità devono essere accompagnate da precisione e qualità, altrimenti i risultati del lavoro potrebbero essere compromessi da un'esperienza scadente causata da funzionalità sviluppate in modo inefficiente o, nella peggiore delle ipotesi, da sistemi instabili e affetti da bug.

Sul fronte della sicurezza, è cruciale limitare i falsi positivi e i falsi negativi, fenomeni che potrebbero rallentare drasticamente un'applicazione generando allarmi di sicurezza privi di fondamento. Una corretta implementazione di DevSecOps richiede che gli strumenti siano in grado di eseguire test di sicurezza per eliminare o ridurre al minimo gli allarmi dovuti a falsi positivi e falsi negativi, fornendo al team di sviluppo informazioni utili per correggere tempestivamente le vulnerabilità individuate.

3.4.1 Jenkins

Jenkins è un server open source, Continuous Integration, Continuous Deployment, scritto in Java.

Jenkins consente di automatizzare le diverse fasi del ciclo di vita del software, dalla compilazione al test alla distribuzione.

Una delle caratteristiche principali di Jenkins è la Jenkins Pipeline che è una suite di plug-in che supporta l'implementazione e l'integrazione di pipeline di distribuzione continua in Jenkins.

Una pipeline di distribuzione continua è un'espressione automatizzata del processo per ottenere il software.

Jenkins Pipeline fornisce un set estendibile di strumenti per modellare pipeline di distribuzione da semplici a complesse.

Le esecuzioni delle pipeline possono essere azionate in vari modi, tra cui un commit oppure ad ogni intervallo di tempo oppure attraverso l'interfaccia web.

Inoltre le sue funzionalità possono essere potenziate dal grande numero di plug-in compatibili, sviluppati e diffusi dalla comunità di utenti e developer che lo usa.

Nel momento in cui si esegue si possono osservare le varie fasi di sviluppo build, test e release per poi passare alla fase di deploy.

In seguito illustreremo delle pipeline di esecuzione in cui ci sarà un fallimento (pipeline senza fallimento vedere sopra).

Stage View

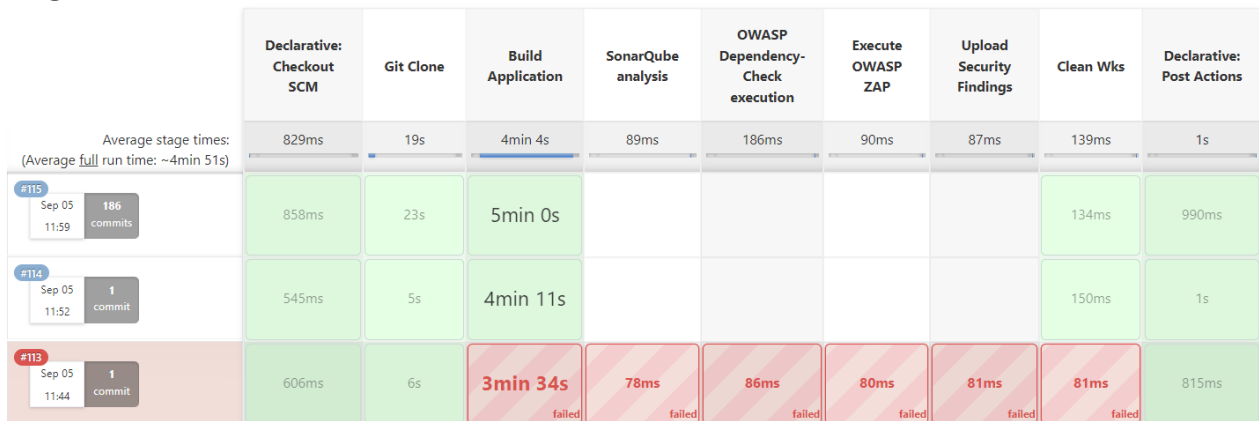


Figura 3.3: Pipeline di fallimento

Da questa immagine si può osservare un fallimento all'interno del processo di build che ha successivamente scatenato un fallimento in tutte le fasi successive. Il processo verrà quindi successivamente scartato. [7]

3.4.2 Snyk

Snyk è una soluzione di sicurezza destinata agli sviluppatori che aiuta le aziende a usare il codice open source e consente di rimanere protetti. Basandosi sul suo esclusivo database di vulnerabilità, Snyk individua e corregge continuamente vulnerabilità note e violazioni delle licenze nelle dipendenze open source. Si integra nel flusso di lavoro degli sviluppatori e negli strumenti di controllo del codice sorgente (ad esempio GitHub, BitBucket, GitLab), agganciandosi alle pipeline CI/CD e monitorando continuamente Platform as a Service (PaaS) e app serverless in produzione.

La scansione altamente accurata di Snyk rileva le vulnerabilità in tempo reale e fornisce un contesto di definizione delle priorità olistico in modo da sapere sempre cosa risolvere per primo.

Snyk segnala ogni vulnerabilità trovata e propone modi di risoluzione del problema.

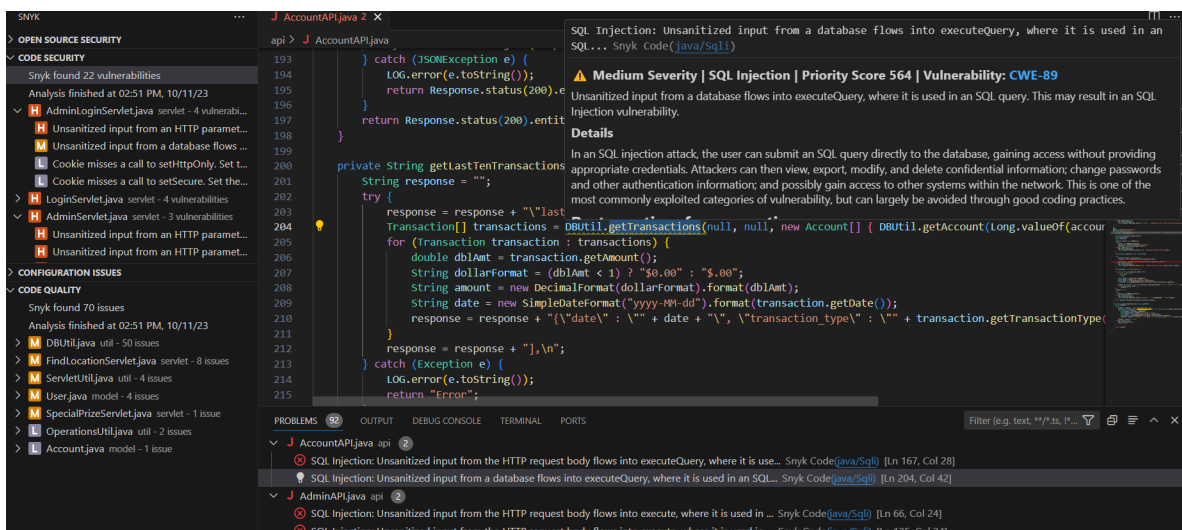


Figura 3.4: Snyk Vulnerabilities

Come possiamo osservare in basso vengono segnalate tutte le vulnerabilità riscontrate. Selezionando una di quelle si aprirà la pagina esatta della vulnerabilità e selezionando il preciso contenuto vulnerabile si potranno visualizzare i dettagli precisi della vulnerabilità ed anche eventuali modi per risolverla. [13]

3.4.3 OWASP Dependency-Check

Dependency check è uno strumento di analisi della composizione software che tenta di rilevare le vulnerabilità contenute nelle dipendenze di un progetto. Il problema è che all'interno delle applicazioni si è soliti includere librerie di terze parti che potrebbero essere vulnerabili.

Come possiamo osservare dalla pipeline di sviluppo è uno degli aspetti da controllare all'interno del ciclo devsecops. [10]

3.4.4 Owasp ZAP

OWASP Zed Attack Proxy (ZAP) è uno strumento di sicurezza che consente di rilevare vulnerabilità in applicazioni e siti Web. È una soluzione semplice e flessibile che si adatta a vari livelli di utilizzo e competenza. ZAP è composto da due macrosezioni. La prima è uno scanner automatizzato di vulnerabilità che consente di identificare problemi e fornisce un report con i dettagli delle vulnerabilità per poter sanare la falla. La seconda permette a ZAP di operare come proxy che consente di ispezionare il traffico e tutta la comunicazione HTTP e gli eventi, con la possibilità di modificarli o analizzare i loro trigger che sono potenzialmente pericolosi per il sistema.

Viene riportato sotto un esempio dello scanner e del report associato ad esso.

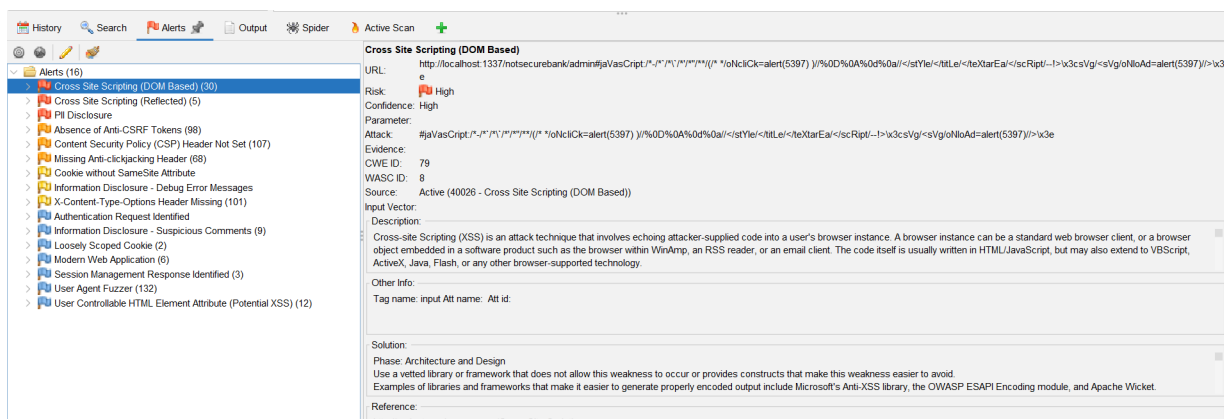


Figura 3.5: Applicazione ZAP

Come possiamo osservare sulla sinistra possiamo trovare la lista di tutte le vulnerabilità trovate, in ordine di criticità. Selezionando una di esse possiamo vedere dove si trova il problema, una descrizione dettagliata del problema ed uno dei modi di risoluzione.

In questo modo una persona che esegue scanning avrà tutte le informazioni necessarie per verificare la problematica ed una volta verificata si potrà classificare come vulnerabilità o come falso positivo.

Nella sicurezza informatica, si parla di falso positivo nel caso in cui si generi un allarme relativo ad un problema di cybersecurity che risulti però infondato. Ovvero nel caso in cui una minaccia viene rilevata e segnalata ma che in realtà non lo è.

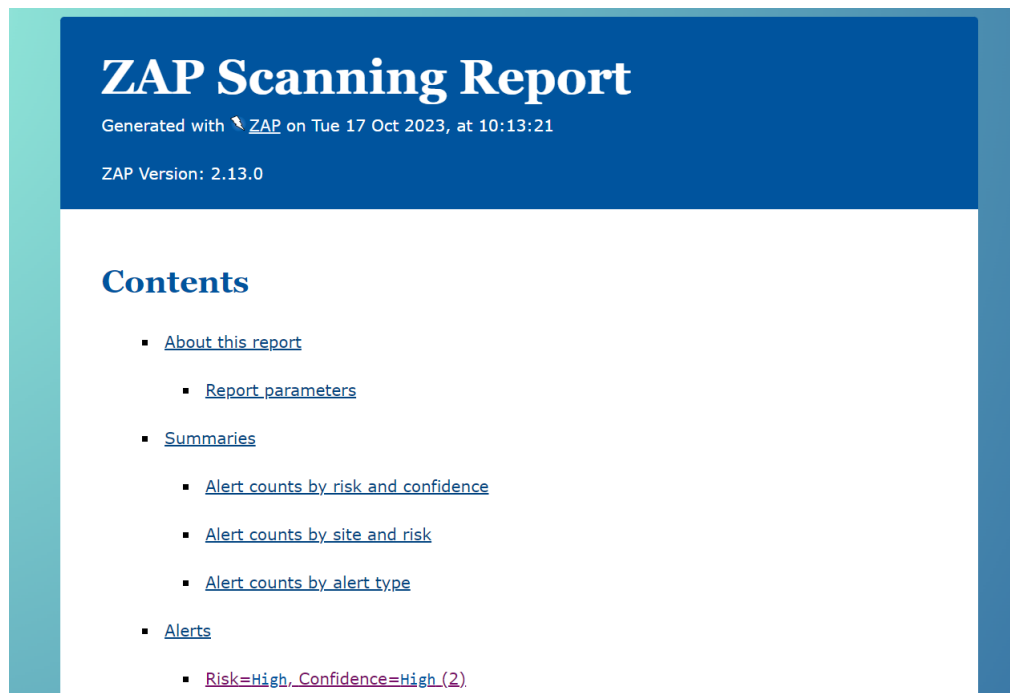


Figura 3.6: ZAP report

Questa è una prima pagina di un report in cui troviamo l'indice di tale documento. Il documento sarà quindi formato da una prima parte introduttiva, una parte che comprende le statistiche e le valutazioni delle vulnerabilità riscontrate ed una parte finale in cui vengono illustrate una ad una le vulnerabilità. In questa ultima parte selezionando la vulnerabilità che si vuole analizzare si potrà trovare un link che porta direttamente alla pagina ufficiale della vulnerabilità OWASP. Offre dunque tutti gli strumenti necessari per poter effettuare una analisi dettagliata e completa delle vulnerabilità. [12]

3.4.5 Esempi pratici

Esempio di test di applicazioni java: Notsecurebank

Motivazioni della scelta del linguaggio: Java è un linguaggio di programmazione orientata a oggetti multiplatforma che viene eseguito su miliardi di dispositivi in tutto il mondo e nonostante sia stato creato più di 20 anni fa, Java è attualmente il linguaggio più diffuso tra gli sviluppatori di App, ne citiamo qui alcune famose come per esempio Spotify, Twitter, LinkedIn e molte altre.

Spesso Java viene utilizzata con un framework come per esempio Spring che è molto diffuso per la creazione di applicazioni autonome, adatte ad ambienti di produzione che vengono eseguite su JVM. Java Spring Boot semplifica e velocizza lo sviluppo di applicazioni web.

Di norma risulta vantaggioso utilizzare lo strumento Maven per i progetti Java, esso è in grado di organizzare le dipendenze facilitando l'organizzazione dei progetti in maniera più pulita standardizzando la sua struttura.

Nel nostro esempio viene utilizzata una semplice applicazione Java.
Strumenti utilizzati:

- Virtual machine NotSecureBank
- OWASP ZAP

Virtual machine NotSecureBank: è la macchina virtuale creata per l'applicazione vulnerabile che si vuole testare.

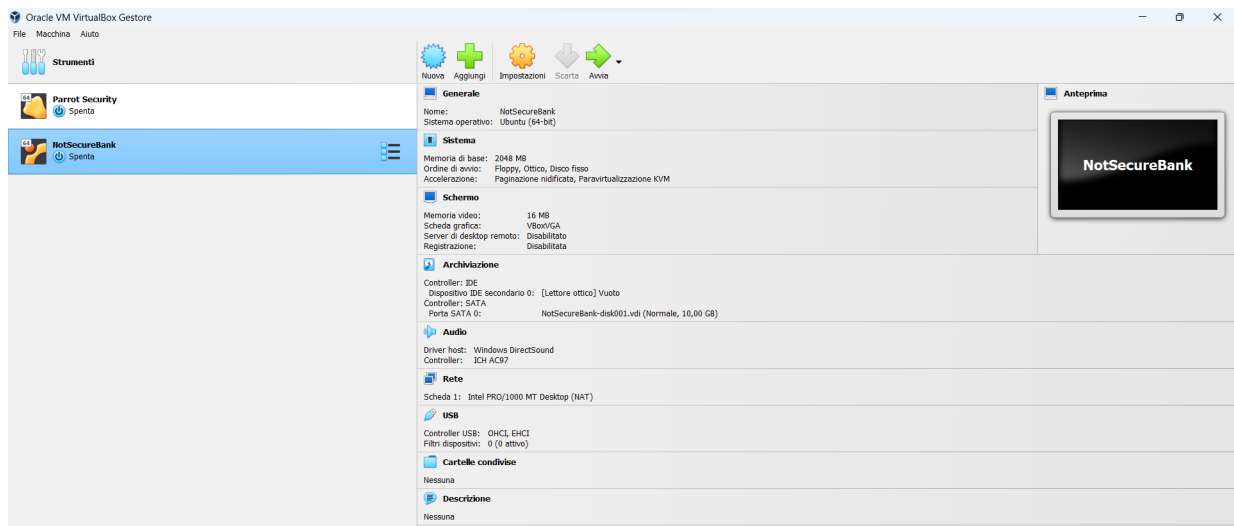


Figura 3.7: Avvio Virtual machine NotSecureBank

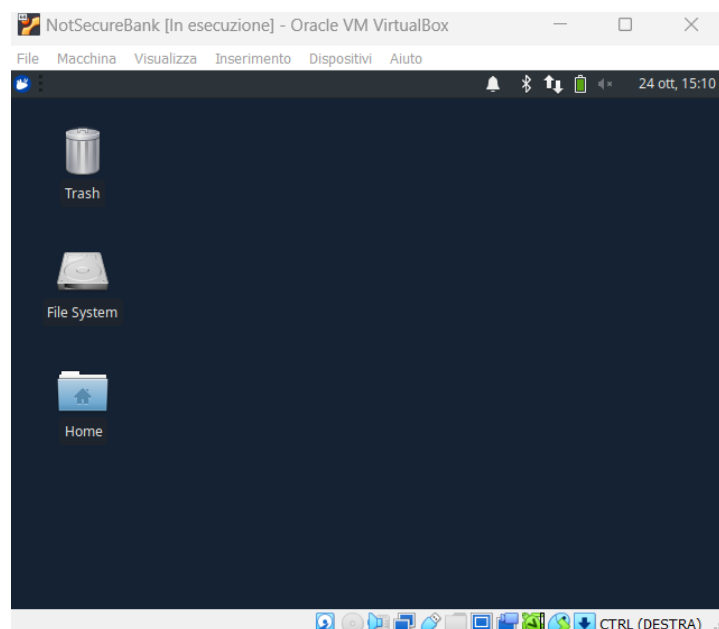


Figura 3.8: Virtual machine NotSecureBank

Una volta avviata la macchina NotSecureBank possiamo fare avviare una esecuzione automatica su Owasp zap all'indirizzo <http://localhost:1337/notsecurebank> a questo punto disponibile.

Viene visualizzato in seguito il risultato della scansione:

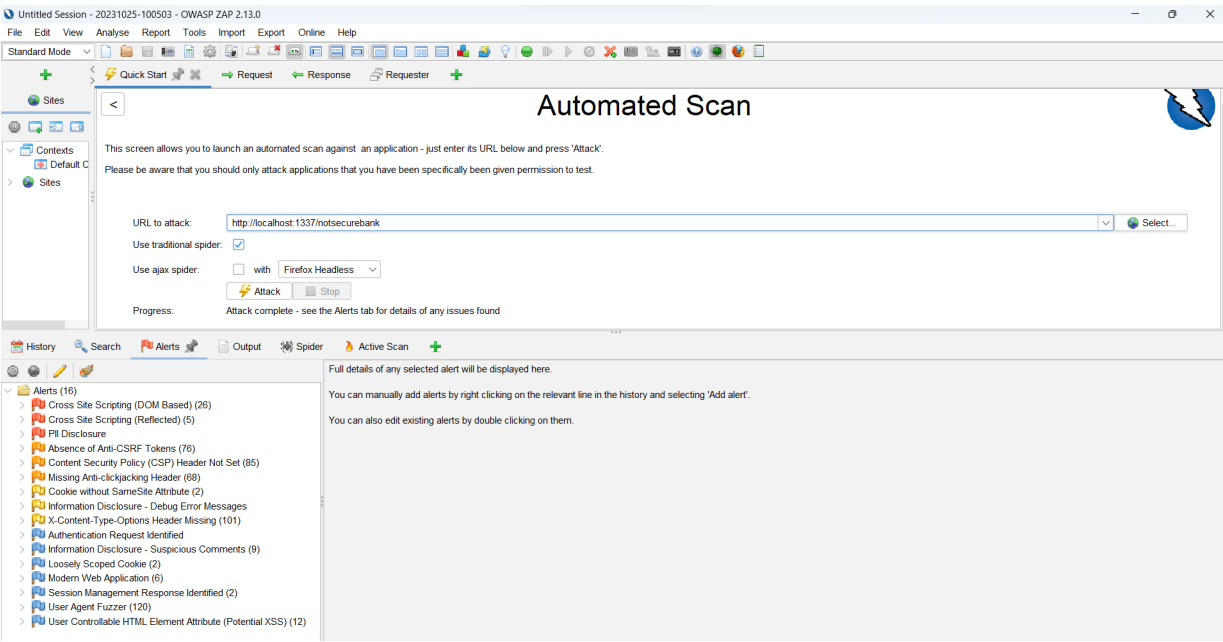


Figura 3.9: Scanner Zap

In basso a sinistra si trovano la lista di vulnerabilità trovate, dalla più grave alla meno grave, e selezionandole si potrà osservare la sua descrizione ed i modi per sanarla. Per generare il report dell’analisi effettuata si dovrà cliccare sul pulsante in alto a destra e si otterrà in maniera automatica un report.

Ecco la visualizzazione delle vulnerabilità all’interno del report:

Alert type	Risk	Count
Cross Site Scripting (DOM Based)	High	26 (162.5%)
Cross Site Scripting (Reflected)	High	5 (31.2%)
PII Disclosure	High	1 (6.2%)
Absence of Anti-CSRF Tokens	Medium	76 (475.0%)
Content Security Policy (CSP) Header Not Set	Medium	85 (531.2%)
Missing Anti-clickjacking Header	Medium	68 (425.0%)
Cookie without SameSite Attribute	Low	2 (12.5%)
Information Disclosure - Debug Error Messages	Low	1 (6.2%)
X-Content-Type-Options Header Missing	Low	101 (631.2%)
Authentication Request Identified	Informational	1 (6.2%)
Information Disclosure - Suspicious Comments	Informational	9 (56.2%)
Loosely Scoped Cookie	Informational	2 (12.5%)
Modern Web Application	Informational	6 (37.5%)
Session Management Response Identified	Informational	2 (12.5%)
User Agent Fuzzer	Informational	120 (750.0%)
User Controllable HTML Element Attribute (Potential XSS)	Informational	12 (75.0%)
Total		16

Figura 3.10: Report finale NotSecureBank

Come possiamo osservare in questo report vengono riportate vere e proprie vulnerabilità e anche varie forme di pericolo, facilmente attaccabili e quindi segnalate.

Questo è solo un indice delle vulnerabilità e dei pericoli trovati, all'interno dell'intero report queste minacce verranno analizzate maggiormente nel dettaglio, descrivendo la vulnerabilità e come risolverla, inserendo riferimenti validi alla documentazione del problema stesso.

Di seguito alla generazione del report è opportuno testare manualmente le vulnerabilità trovate.

Esempio di test in angular

Motivazioni sulla scelta del linguaggio: Angular è un framework di sviluppo web ampiamente utilizzato e apprezzato, che offre una vasta gamma di funzionalità e vantaggi per le aziende che desiderano creare applicazioni web dinamiche e interattive. Offre inoltre un ambiente di sviluppo solido e sicuro, riducendo al minimo gli errori e migliorando la manutenibilità del codice. Eccone alcuni esempi: Netflix, PayPal e molte altre. Oltre ad Angular possono essere utilizzate delle tecnologie analoghe per modalità d'uso e funzionalità, le più utilizzate sono React e Vue.

Questi framework o librerie, sono utilizzabili dopo l'installazione di Node.js che comprende NPM (Node Package Manager), il quale fornisce le dipendenze del progetto tramite configurazione, eventualmente riscontrando se ci sono vulnerabilità note all'interno del progetto.

Ecco un esempio dell'installazione delle dipendenze del progetto:

```
npm install

up to date, audited 1104 packages in 5s

132 packages are looking for funding
  run 'npm fund' for details

2 vulnerabilities (1 moderate, 1 high)

To address all issues possible, run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.

Run 'npm audit' for details.
```

Figura 3.11: Installazione npm

Si può osservare che (a differenza di un'installazione attraverso Maven) sono già state segnalate 2 vulnerabilità. Per effettuare un'analisi delle vulnerabilità si utilizza il comando `npm audit`.

```
# npm audit report

prismjs <=1.24.1
Severity: high
Cross-Site Scripting in Prism - https://github.com/advisories/GHSA-wvwm-4hhf-97x9
prismjs Regular Expression Denial of Service vulnerability - https://github.com/advisories/GHSA-hqhp-5p83-hx96
Regular Expression Denial of Service (ReDoS) in Prism - https://github.com/advisories/GHSA-gj77-59wh-66hg
Denial of service in prismjs - https://github.com/advisories/GHSA-h4hr-7fg3-h35w
fix available via `npm audit fix --force`
Will install prismjs@1.29.0, which is outside the stated dependency range
node_modules/prismjs

quill <=1.3.7
Severity: moderate
Cross-site Scripting in quill - https://github.com/advisories/GHSA-4943-9vvg-gr5r
No fix available
node_modules/quill

2 vulnerabilities (1 moderate, 1 high)

To address all issues possible, run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.
```

Figura 3.12: npm audit

```
# npm audit report

ecstatic <=4.1.2
Severity: high
Denial of Service in ecstatic - https://github.com/advisories/GHSA-jc84-3g44-wf2q
Open Redirect in ecstatic - https://github.com/advisories/GHSA-9q64-mpxx-87fg
Denial of Service in ecstatic - https://github.com/advisories/GHSA-pm9p-9926-w68m
fix available via `npm audit fix --force`
Will install http-server@14.1.1, which is a breaking change
node_modules/ecstatic
  http-server 0.4.0 - 0.12.1
  Depends on vulnerable versions of ecstatic
  Depends on vulnerable versions of optimist
  Depends on vulnerable versions of union
  node_modules/http-server

json5 <1.0.2
Severity: high
Prototype Pollution in JSON5 via Parse Method - https://github.com/advisories/GHSA-9c47-m6qq-7p4h
fix available via `npm audit fix --force`
Will install babelify@10.0.0, which is a breaking change
node_modules/json5
  babel-core 5.8.20 - 7.0.0-beta.3
  Depends on vulnerable versions of babel-register
  Depends on vulnerable versions of json5
  node_modules/babel-core
    babel-register *
    Depends on vulnerable versions of babel-core
    node_modules/babel-register
  babelify 7.0.0 - 7.3.0
```

Figura 3.13: Visualizzazione di vulnerabilità

Strumenti utilizzati:

- Docker Desktop
- Progetto github HackMEAN
- OWASP zap

Docker Desktop: Docker è un popolare software libero progettato per eseguire processi informatici in ambienti isolabili, minimali e facilmente distribuibili chiamati container Linux, con l'obiettivo di semplificare i processi di deployment di applicazioni software. Utilizzando i container le risorse possono essere isolate, i servizi

limitati e i processi avviati in modo da avere una prospettiva completamente privata del sistema operativo, col loro proprio identificativo, file system e interfaccia di rete. I vantaggi di Docker si misurano in relazione alle macchine virtuali. Infatti, i container sono più leggeri delle macchine virtuali, vengono avviati più velocemente e richiedono meno risorse.

Progetto github HackMEAN: è una applicazione intenzionalmente vulnerabile costruita sullo stack MEAN (MongoDB, Express, Angular e Node.js). Questa applicazione illustra le vulnerabilità comuni nello stack MEAN. Il caso studio permette di trovare tutte le vulnerabilità eseguendo l'applicazione in uno scanner automatico e generarne il report associato.

Con l'utilizzo di docker, una volta scaricato il progetto ed aggiunto ai container, è possibile avviare l'applicazione e una istanza mongodb in una volta sola.

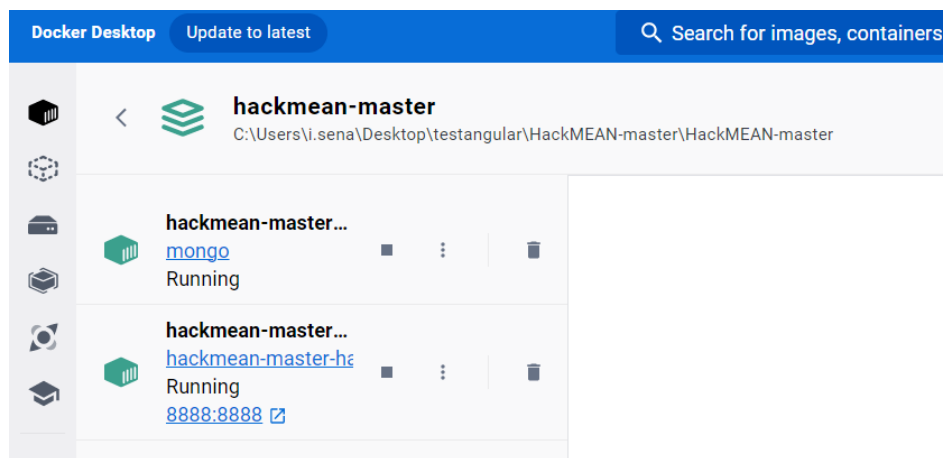


Figura 3.14: Visualizzazione applicazione in Docker

In seguito verrà illustrata l'applicazione in esecuzione:

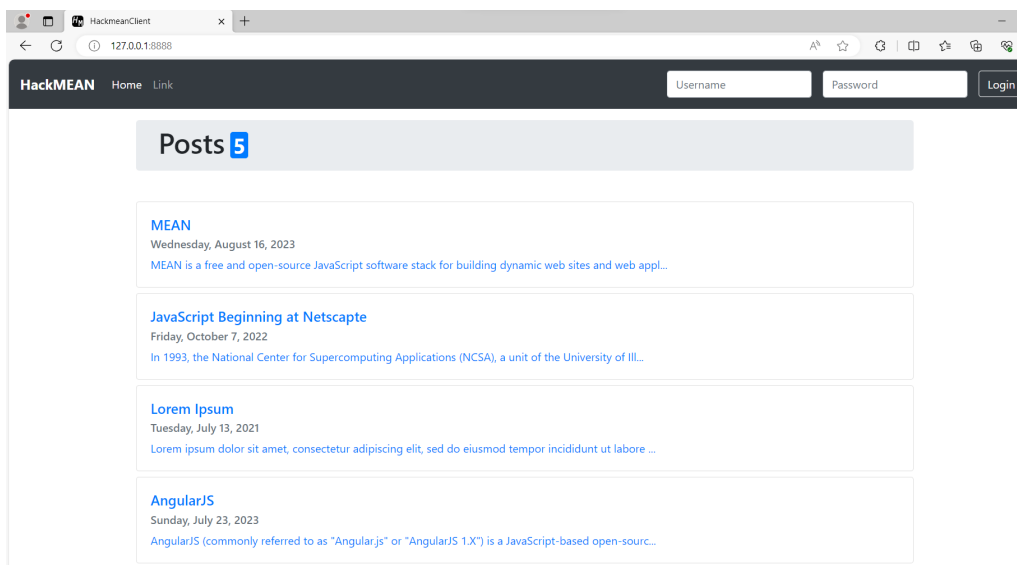


Figura 3.15: Applicazione HackMEAN

Con la visualizzazione del sito dell'applicazione possiamo ricavarne il link, che in seguito utilizzeremo per effettuare una scansione automatizzata su OWASP zap.

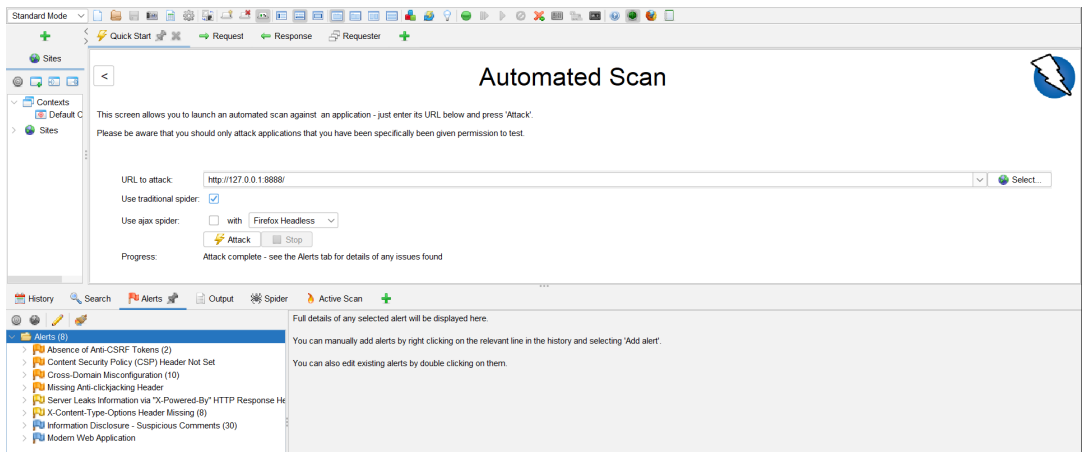


Figura 3.16: Scanner HackMEAN

In basso a sinistra si trovano la lista di vulnerabilità trovate, dalla più grave alla meno grave, e selezionandole si potrà osservare la sua descrizione ed i modi per sanarla. Per generare il report dell’analisi effettuata si dovrà cliccare sul pulsante in alto a destra e si otterrà in maniera automatica un report. Ecco la visualizzazione delle vulnerabilità all’interno del report:

Alert type	Risk	Count
Absence of Anti-CSRF Tokens	Medium	2 (25.0%)
Content Security Policy (CSP) Header Not Set	Medium	1 (12.5%)
Cross-Domain Misconfiguration	Medium	10 (125.0%)
Missing Anti-clickjacking Header	Medium	1 (12.5%)
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	10 (125.0%)
X-Content-Type-Options Header Missing	Low	8 (100.0%)
Information Disclosure - Suspicious Comments	Informational	30 (375.0%)
Modern Web Application	Informational	1 (12.5%)
Total		8

Figura 3.17: Report HackMEAN

Come possiamo osservare non sempre nei report vengono riportate vere e proprie vulnerabilità ma spesso si trovano potenziali forme di pericolo, facilmente attaccabili e quindi segnalate dal report. Questo è solo un indice dei pericoli trovati, all’interno dell’intero report queste minacce verranno analizzate maggiormente nel dettaglio, descrivendo la vulnerabilità e come risolverla, inserendo riferimenti validi alla documentazione del problema stesso. Di seguito alla generazione del report è opportuno testare manualmente le vulnerabilità trovate.

Capitolo 4

Minacce Cloud note e risoluzioni

4.0.1 Vulnerabilità comuni in applicazioni Cloud

Ecco di seguito elencate alcune delle vulnerabilità cloud più comuni:

- **Configurazione errata nei contenitori:** Una delle vulnerabilità più comuni è un'errata configurazione dei container che ha portato a gravi violazioni della piattaforma cloud. Queste configurazioni errate sono problemi che si sono verificati a causa della mancanza delle impostazioni sicure richieste o dell'accesso pubblico alla piattaforma cloud. L'errata configurazione del gruppo di sicurezza è un altro tipo di vulnerabilità che si verifica in un sistema di sicurezza e porta all'accesso diretto alla piattaforma cloud.
- **Autenticazione impropria:** Un processo di autenticazione inadeguato nel sistema di sicurezza cloud è un'altra vulnerabilità comune che spesso porta a violazioni della sicurezza. La mancanza di autenticazione a più fattori e la scarsa sicurezza della password rappresentano alcune gravi sfide nella gestione delle vulnerabilità del cloud. Inoltre, in assenza di un adeguato controllo degli accessi, qualsiasi utente autorizzato a cui non dovrebbe essere consentito l'accesso potrebbe riuscire ad accedere al sistema cloud.
- **Inadempienza:** La non conformità agli standard di settore come PCI-DSS, HIPAA, ISO 27001 e SOC 2 è una delle principali cause di vulnerabilità nella piattaforma cloud. Queste policy di gestione delle vulnerabilità cloud garantiscono che il carico di lavoro cloud sia adeguatamente protetto. Quando il fornitore di sicurezza cloud e il cliente gestiscono in modo improprio le impostazioni di sicurezza, si verificano problemi e gli aggressori ne approfittano per ottenere l'accesso al server.
- **Configurazione di rete errata:** È fondamentale che la rete cloud sia configurata adeguatamente; in caso contrario, rischierebbe la sicurezza dei dati. Il team di sicurezza dovrebbe fare attenzione nel fornire l'accesso alla porta delle reti private perché se viene lasciata aperta per impostazione predefinita, chiunque può accedere ai dati sensibili. Il team di sicurezza dovrebbe fornire l'accesso alle porte specifiche necessarie per l'accesso e tutte le altre porte dovrebbero essere disabilitate per la connessione alle reti private.
- **Scarsa gestione degli accessi ai dati sensibili:** le piattaforme Cloud contengono molte applicazioni specifiche che possono aiutare chiunque ad accedere a molti

dati accessibili. Pertanto è fondamentale non creare mai credenziali o ID di gruppo per tutte le applicazioni. Sarebbe l'ideale creare credenziali specifiche per applicazioni specifiche in cui solo il personale autorizzato potrà accedere.

- API protette in modo improprio: le API non sicure rappresentano uno dei modi principali con cui gli aggressori possono accedere alle piattaforme Cloud e rubare tutti i dati essenziali. Gli aggressori cercano API prive di autorizzazione e autenticazione adeguate e le sfruttano per scopi illeciti.

A differenza delle vulnerabilità web non ci sono ancora delle vere e proprie classificazioni.

4.0.2 Pratiche di gestione delle vulnerabilità del Cloud

La gestione delle vulnerabilità nel cloud computing può essere definita come un approccio o processo continuo volto a identificare, analizzare, dare priorità e risolvere i problemi di sicurezza nel cloud computing. Non solo riduce al minimo eventuali rischi per la sicurezza correggendo le vulnerabilità comuni, ma garantisce anche che tutte le lacune di sicurezza attraverso le quali gli aggressori possono entrare nel framework siano adeguatamente bloccate. Il cloud computing moderno è così vasto e complesso che componenti come dispositivi IoT, firewall di configurazione e contenitori di carichi di lavoro potrebbero causare problemi di sicurezza per questo risulta molto importante applicare le seguenti pratiche per garantire una sicurezza maggiore:

- Scansione costante delle vulnerabilità del cloud: utilizzare la capacità di scansione continua delle vulnerabilità del cloud per scansionare e rilevare le vulnerabilità nella loro fase iniziale. Questi scanner dovrebbero essere supportati da un elenco completo e aggiornato di vulnerabilità in modo che sia in grado di rilevare le minacce.
- Test di penetrazione sistematico: eseguire test di penetrazione aiuta ad analizzare operativamente il danno che può verificarsi in caso di attacchi.
- Scansione delle vulnerabilità durante l'integrazione: l'utilizzo della scansione continua durante la fase di sviluppo di un'applicazione è un modo per mantenere una protezione completa.

Queste risultano essere le best practice d'eccellenza per gestire le vulnerabilità in applicazioni Cloud, certamente questo non vuole essere un elenco esaustivo perché la gestione delle vulnerabilità dipendono oltre che da questi fattori anche dalla tipologia di applicazione che si vuole creare. [8]

Capitolo 5

Conclusioni finali

Durante il corso di questo studio, è emersa chiaramente l'importanza della sicurezza informatica nel contesto della nostra vita quotidiana, caratterizzata da una stretta interazione con la tecnologia. Questo percorso ha rappresentato una significativa presa di coscienza riguardo alla necessità di proteggere le nostre informazioni e i nostri sistemi da minacce sempre più sofisticate e diffuse.

Durante questa ricerca, ho avuto l'opportunità di esaminare numerose vulnerabilità comuni e di acquisire competenze essenziali per riconoscerle e affrontarle in modo efficace. Inoltre, ho approfondito l'utilizzo di strumenti ampiamente utilizzati per testare la sicurezza delle applicazioni, imparando anche ad analizzare e risolvere le vulnerabilità individuate.

L'esperienza presso l'azienda mi ha permesso di osservare e comprendere le metodologie adottate dall'azienda stessa per valutare i rischi e gestire le pratiche di sicurezza.

La continua evoluzione delle applicazioni nell'ambiente aziendale mi ha offerto inoltre l'opportunità di approfondire gli aspetti della sicurezza legati al cloud computing e di familiarizzare con le best practice più comuni per affrontare la gestione delle vulnerabilità in questo contesto dinamico e complesso.

Durante il corso di questa ricerca, è stato fondamentale garantire che tutte le analisi e le metodologie adottate fossero quanto di più attuali e pertinenti possibile.

È importante sottolineare che il settore della sicurezza informatica è in continua evoluzione, con nuove minacce e vulnerabilità che emergono costantemente. Pertanto, le competenze e le pratiche acquisite durante questa ricerca sono state concepite come una base solida, ma al contempo flessibili e adattabili alle sfide future.

Bibliografia

- [1] AVG. Attacchi zero-day. <https://www.avg.com/it/signal/zero-day-attack>. Accessed: 2023-11-10.
- [2] AWS. Cos'è devsecops? <https://aws.amazon.com/it/what-is/devsecops/>. Accessed: 2023-11-10.
- [3] G. Butti, A. Di Mattia, I. Gabrielli, P. Giudice, F. Livelli, C. Mauceli, Y. Perseu, L. Querzoni, S. Scozzari, and A. Manzoni. Rapporto clusit sulla sicurezza informatica. 2022. Accessed: 2023-11-10.
- [4] cloudflare. Definire un attacco ddos. <https://www.cloudflare.com/it-it/learning/ddos/what-is-a-ddos-attack/>. Accessed: 2023-11-10.
- [5] cybersecurity360. Allerta per una zero-day su cisco ios xe. <https://www.cybersecurity360.it/nuove-minacce/allerta-per-una-zero-day-su-cisco-ios-xe-oltre-1-600-i-dispositivi-di-rete-esposti-in-italia/>. Accessed: 2023-11-10.
- [6] cybersecurity360. Autenticazione a due fattori. Accessed: 2023-11-10.
- [7] geekandjob. Jenkins. <https://www.geekandjob.com/wiki/jenkins>. Accessed: 2023-11-10.
- [8] Docs Italia. Manuale di abilitazione al cloud. 2022. Accessed: 2023-11-10.
- [9] malwarebytes. Malware. <https://it.malwarebytes.com/malware/>. Accessed: 2023-11-10.
- [10] OWASP. Controllo delle dipendenze owasp. <https://owasp.org/www-project-dependency-check/>. Accessed: 2023-11-10.
- [11] OWASP. Owasp top 10. <https://owasp.org/Top10/it/>. Accessed: 2023-11-10.
- [12] OWASP Zed Attack Proxy. Owasp zed attack proxy. <https://oscat.regione.toscana.it/catalogo-prodotti-open-source/vulnerability-assessment-web-app/owasp-zed-attack-proxy>. Accessed: 2023-11-10.
- [13] trend. Snysk. https://www.trendmicro.com/it_it/snyk.html. Accessed: 2023-11-10.
- [14] UniverseIT. Devsecops? <https://universeit.blog/devsecops-development-security-operations/>. Accessed: 2023-11-10.

Vorrei dedicare questa sezione a tutti coloro che mi hanno accompagnata e supportata in questo percorso.

Per prima cosa ci tengo a ringraziare la mia mamma per esserci sempre stata, per avermi sempre sostenuta, incoraggiata ed amata. Grazie per essere la mia mamma e la mia migliore amica, per avermi ascoltata e per avermi dato forza sempre, sei e sarai sempre indispensabile per me.

Grazie al mio papà per essere sempre stato una guida. Nonostante le nostre divergenze rimarrai sempre per me un grande punto di riferimento, ti voglio bene.

Grazie a tutta la mia meravigliosa famiglia per avermi sempre sostenuta.

Un grazie tanto speciale a Ricca per aver sempre creduto in me, per avermi sempre tranquillizzata e per avermi fatta sentire capita e meno sola in questo percorso.

Grazie anche a Fede e Cla, i miei amici del cuore, per esserci sempre stati e per avermi strappato sempre un sorriso in qualsiasi occasione.

Un grande grazie a Samu, mio tutor e ormai amico, per avermi fatto vivere una bellissima esperienza, senza te non sarebbe stato possibile.

Grazie anche a tutte le splendide persone che ho incontrato durante il mio stage, in particolare Gabri e Gas, avete reso questa esperienza migliore, ed ho trovato in voi due bellissime persone, senza di voi sicuramente non sarebbe stato lo stesso.

Un grazie anche a tutte quelle persone che sono state parte di questo viaggio, ognuno di voi a suo modo ha lasciato un bellissimo ricordo in me.