



UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

TESI DI LAUREA

DevSecOps: Problemi di sicurezza e soluzioni in una
CI/CD pipeline per un'applicazione cloud critica

RELATORE:

Prof. Salvatore La Torre

RELATORI ESTERNI:

Claudio Carotenuto

Giuseppe Amato

CANDIDATO:

Alfonso Ingenito

Matricola: 0512104959

ANNO ACCADEMICO 2018-2019

Abstract

La velocità e il continuo cambiamento sono aspetti che caratterizzano la società odierna, così come la competitività delle aziende all'interno del mercato. Tra le diverse metodologie di sviluppo software nate per soddisfare queste esigenze ci concentreremo su DevOps, metodologia allo stato dell'arte che garantisce un'elevata velocità di sviluppo e rilascio del prodotto, uno stretto contatto con il cliente e un abbattimento delle divisioni tra team di sviluppo (Dev) e sistemisti (Ops). Se da un lato l'abbattimento delle barriere tra gli sviluppatori e i clienti apporta numerosi vantaggi, dall'altro introduce nuove problematiche di sicurezza e aggrava gli impatti di quelle già note. La sicurezza in ambienti di sviluppo caratterizzati da una elevata velocità di rilascio, come accade con DevOps, è quindi divenuta di massima importanza per le aziende, portandola ad essere oggetto di ricerca accademica e campo di sperimentazione di nuove tecnologie.

In questa tesi verranno analizzati i principali aspetti di sicurezza dello sviluppo con DevOps e individuate le possibili soluzioni, tra cui l'applicazione di una nuova metodologia, DevSecOps, che pone l'attenzione alla sicurezza già dalle prime fasi dello sviluppo software, avvalendosi di strumenti automatizzati per non rallentare il ciclo di vita. Inizialmente verrà effettuata una definizione di DevOps, effettuando anche una descrizione degli strumenti utilizzati e delle problematiche di sicurezza di tale metodologia. Verranno poi identificati e descritti i relativi metodi di mitigazione, avvalendosi di un threat model. Le soluzioni individuate saranno quindi sperimentate in ambiente reale con l'implementazione della metodologia DevSecOps a partire da un sistema virtualizzato basato su DevOps. Queste operazioni permetteranno di valutare l'efficacia degli strumenti utilizzati nonché di mitigare le problematiche riscontrate. Lo scopo ultimo sarà quindi di capire quanto queste metodologie consentano alle grandi aziende di ridurre i rischi legati alla sicurezza mantenendo sostenibili i costi e la velocità di sviluppo dei propri prodotti.

Sommario

Introduzione 1

Capitolo 1: DevOps 5

1.1. Definizione 5

1.2. Metodo Agile 6

1.3. I vantaggi di DevOps 7

1.4. Tecnologie..... 9

1.4.1. Jenkins.....9

1.4.2. Blue Ocean.....12

1.4.3. Git e GitHub.....14

Capitolo 2: Problemi di security in DevOps 16

2.1. Problematiche della metodologia 16

2.2. Vulnerabilità del software 17

2.3. Catalogazione delle vulnerabilità 20

2.4. Threat modeling 22

Capitolo 3: Mitigazione delle problematiche..... 27

3.1. Principi di sicurezza 27

3.2. DevSecOps 29

3.2.1. Definizione.....29

3.2.2. Sicurezza dell'ambiente e dei dati.....30

3.2.3. Sicurezza del processo CI/CD.....31

Capitolo 4: Sperimentare DevSecOps come soluzione ai problemi di sicurezza in una pipeline di CI/CD. 33

4.1. Test di sicurezza 34

4.1.1. Tipologie di Attacchi.....34

4.1.2. Testing con OWASP ZAP.....36

4.1.3. Test con Nessus.....	40
4.1.4. Threat analysis.....	43
4.2. Hardening di Jenkins.....	48
4.2.1. Configurazione di LDAP.....	48
4.2.2. Definizione dei ruoli nel sistema.....	52
4.2.3. Controllo del codice.....	53
4.2.4. Ulteriori configurazioni di sicurezza.....	58
4.3. Valutazione dei rischi residui.....	59
Capitolo 5: Conclusioni e sviluppi futuri.....	60

Introduzione

La velocità e il continuo cambiamento sono aspetti che caratterizzano la società odierna, così come la competitività delle aziende all'interno del mercato. Al giorno d'oggi è diventato inoltre di vitale importanza per un'azienda produttrice di software cogliere rapidamente le opportunità di mercato, ottimizzare la raccolta dei feedback del cliente e ridurre al minimo i costi dovuti al tempo impiegato per la produzione.

In risposta alle necessità del mercato sono state realizzate delle metodologie di sviluppo software che si focalizzano sulla rapidità di sviluppo e di rilascio di nuove funzionalità, tra queste la metodologia Agile e DevOps. La prima di queste è basata su un manifesto che raccoglie diversi principi i quali si contrappongono ai tradizionali modelli di sviluppo e hanno come obiettivi principali la piena soddisfazione del cliente, l'abbattimento dei costi e dei tempi di sviluppo del software e l'aumento della qualità di quest'ultimo. Ci concentreremo in questa tesi su DevOps, metodologia di sviluppo software allo stato dell'arte, che applica i principi posti da Agile migliorandone molti aspetti tra cui l'estrema flessibilità, focalizzandosi principalmente sul prodotto richiesto dal cliente. Grandi aziende come Ericsson hanno deciso di applicare questa metodologia per ottenerne i molteplici benefici economici legati a questa innovazione. Tali benefici sono bilaterali, sia per l'azienda che per il mercato. In ambito aziendale i benefici si quantificano in una ottimizzazione dei costi, in quanto sussiste una riduzione del time-to-market, e un aumento dei ricavi generato da una maggiore possibilità di soddisfazione della richiesta. In termini di mercato, l'implementazione di DevOps permette un incremento della produzione a costi più bassi che si traduce in prezzi concorrenziali del prodotto, beneficiati dal cliente.

Il cuore di DevOps è l'abbattimento delle barriere culturali e di competenze solitamente esistenti tra i team di sviluppo (Dev) e quelli operativi (Ops) facendo uso delle pratiche di Continuous Integration (CI) e Continuous Deployment (CD) del prodotto, ovvero: le funzionalità sviluppate in piccole parti per volta, vengono

continuamente integrate (CI) e rilasciate (CD), avvalendosi di una elevata automazione dei processi, ottenuta tramite l'utilizzo di numerosi tool, illustrati in questa tesi. Tutto questo porta ad un incremento della frequenza dei rilasci che risulta di vitale importanza nelle metodologie "agili" poiché consente di ottenere un feedback rapido sulla qualità del prodotto percepita dall'utente. Si pensi ad esempio che Amazon, tra le prime aziende a adottare questa metodologia, nel 2013 rilasciava una nuova versione del proprio software ogni 11.6 secondi, per poi passare ad un rilascio di 50 milioni di aggiornamenti annuali nel 2015 [1]. Secondo le tempistiche appena mostrate è possibile affermare che durante l'acquisto di un insieme di prodotti sul sito web amazon.it, in maniera totalmente trasparente, si venga serviti da differenti versioni della piattaforma software, la quale risulta essere migliore di volta in volta.

Se da un lato l'abbattimento delle barriere tra gli sviluppatori e i clienti apporta i numerosi vantaggi discussi sopra, dall'altro introduce nuove problematiche di sicurezza e aggrava gli impatti di quelle già note. In Ericsson questi aspetti sono particolarmente sentiti, soprattutto in ottica 5G dove possono rappresentare un vero e proprio impedimento alla diffusione dei nuovi scenari che esso abilita [2]. Si pensi ad esempio, ad un dispositivo IoT di tipo medicale che invia dati sulla salute di un paziente attraverso una rete di telecomunicazioni. L'introduzione, involontaria o meno, di una falla di sicurezza da parte degli sviluppatori, raggiungerebbe istantaneamente il dispositivo, provocando danni diretti come il furto di informazioni sensibili, malfunzionamenti del dispositivo a danno della salute del paziente, o danni indiretti come l'introduzione di vulnerabilità software attaccabili poi da hacker. Come conseguenza dei danni agli utenti il tutto porta spesso ad una rilevante perdita economica e d'immagine per l'azienda produttrice del software.

La sicurezza in ambienti di sviluppo caratterizzati da una elevata velocità di rilascio, come accade con DevOps, è quindi divenuta di massima importanza per le aziende, portandola ad essere oggetto di ricerca accademica e campo di sperimentazione di nuove tecnologie.

Verranno analizzati i principali aspetti di sicurezza di un ambiente di sviluppo basato su DevOps, progettato per la creazione di applicazioni critiche su cloud. Si evidenzieranno le vulnerabilità riscontrate, individuandone poi le possibili

soluzioni, tra cui l'applicazione della nuova metodologia DevSecOps. Questa metodologia, sviluppata a partire da DevOps ha come obiettivo l'integrazione, già dalle prime fasi dello sviluppo software, degli strumenti per il controllo della sicurezza [3]. DevSecOps permette inoltre di mantenere un buon rapporto tra velocità di rilascio e mitigazione delle vulnerabilità, grazie all'automatizzazione dei processi di identificazione dei rischi, introdotta dagli strumenti utilizzati in fase di sviluppo. L'automatizzazione in questione permette di aumentare il livello di sicurezza del sistema, grazie ai report forniti in tempo reale dopo ogni test eseguito prima del rilascio dell'applicazione. I risultati forniti dai software automatici, essendo molto dettagliati, permettono allo sviluppatore di correggere velocemente la problematica riscontrata con una precisione maggiore e senza sacrificare grandi quantità di tempo per il rilascio. L'automatizzazione permette inoltre una riduzione del personale da adoperare per effettuare i test manualmente, che insieme agli altri vantaggi apportati, permette di ottimizzare i costi per l'azienda.

Lo scopo ultimo sarà quindi di capire quanto queste metodologie e questi strumenti consentano alle grandi aziende di ridurre i rischi legati alla sicurezza mantenendo sostenibili i costi e la velocità di sviluppo dei propri prodotti.

Il resto della tesi è organizzato come segue.

Nel primo capitolo verranno approfonditi diversi aspetti di DevOps, inizialmente tramite una sua definizione e la descrizione delle sue origini, riconducibili alla metodologia Agile di cui verrà descritto il manifesto che mette in chiaro suoi principi e i vantaggi apportati dall'utilizzo di Agile. Successivamente verranno trattati i vantaggi apportati da DevOps nelle aziende che la adottano e infine le tecnologie tipicamente utilizzate per implementare la metodologia.

Il secondo capitolo sarà incentrato sulle problematiche di sicurezza introdotte da DevOps, le quali rappresentano il problema principale da dover risolvere. Si parlerà poi del Threat modeling, diagramma utilizzato per avere una visione dall'alto del sistema atta a rivedere tutti gli aspetti di sicurezza e riscontrare le sue vulnerabilità. Descriveremo infine delle metodologie standard utilizzate per catalogare le vulnerabilità, utili per comprendere il processo della loro mitigazione.

Il terzo capitolo sarà incentrato sulle tecniche di mitigazione delle problematiche, precedentemente presentate, descrivendo i principi su cui si basa la sicurezza

informatica e definendo DevSecOps, metodologia di sviluppo software tuttora oggetto di ricerca scientifica e improntata sulla sicurezza applicata in ogni passo della catena di sviluppo. Verrà proposta l'applicazione di DevSecOps come possibile soluzione alle problematiche riscontrate in DevOps.

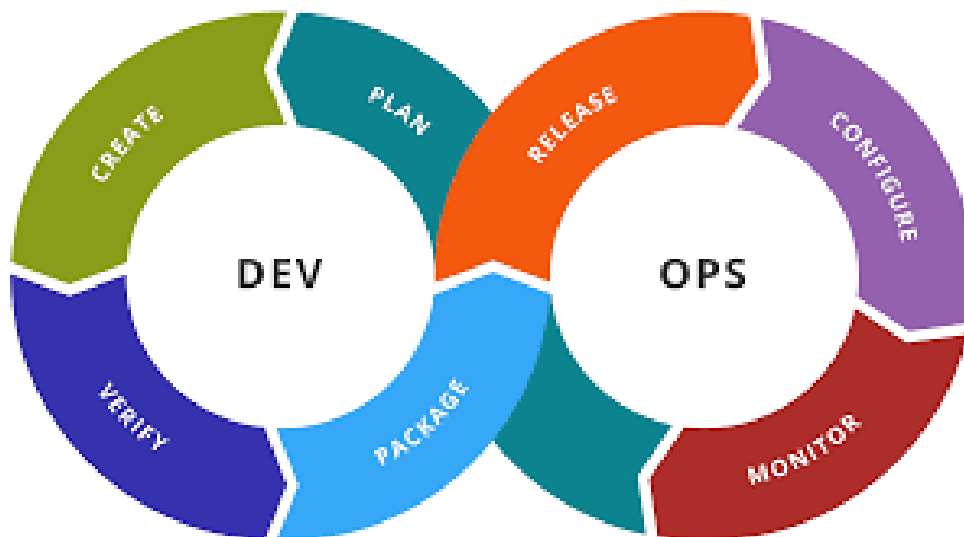
Nel quarto capitolo verranno descritte le attività effettuate per applicare DevSecOps su di un ambiente virtualizzato che adotta la metodologia DevOps. In particolare, verrà descritta una prima fase riguardante i test di sicurezza effettuati sull'attuale sistema. In questa fase verranno prima descritte le tipologie di attacchi che si possono ricevere e, successivamente, si approfondiranno i diversi tool di sicurezza adoperati per rivelare le vulnerabilità presenti nell'ambiente DevOps, descrivendone il loro utilizzo e i risultati ottenuti. Successivamente, in una seconda fase, verranno descritte le attività atte ad effettuare l'hardening del sistema, ovvero, si descriverà una sequenza di operazioni atte a mitigare alcune delle problematiche ritenute di maggior impatto per il livello generale di sicurezza del sistema. Il capitolo si concluderà riportando le problematiche di sicurezza restanti all'interno del sistema, lasciate poiché considerate di scarso impatto per il sistema.

Il quinto ed ultimo capitolo conterrà le conclusioni tratte da questa esperienza, i risultati ottenuti, i possibili sviluppi futuri.

Capitolo 1: DevOps

In questo capitolo verrà descritto DevOps, a partire dalla sua definizione, passando ad una descrizione del manifesto Agile, che espone il pensiero dell'omonima metodologia da cui deriva DevOps. Successivamente verranno presentate le problematiche di sicurezza introdotte da DevOps da tenere in considerazione nel contesto delle grandi aziende. Infine, verranno trattate le tecnologie utilizzate per applicare questa metodologia.

1.1. Definizione



DevOps è una metodologia di sviluppo software che promuove la comunicazione, collaborazione e integrazione tra i team di sviluppatori (developers), che realizzano un servizio, e il team operativo (operations), che mette a disposizione del cliente il servizio [4]. Il nome DevOps deriva proprio dalla fusione dei due nomi “developers” e “operations”.

In DevOps sussiste il concetto di pipeline, inteso come una sequenza esatta di passi che vengono eseguiti per la realizzazione del software. Questa metodologia è composta da due pipeline, la prima chiamata “Delivery Pipeline”, verso il cliente, utilizzata per lo sviluppo, testing e rilascio della funzionalità. La seconda pipeline è chiamata “Feedback loop”, proveniente dal

cliente ed utilizzata per monitorare l'applicazione dopo il rilascio. Il monitoraggio dell'applicazione viene effettuato mediante feedback, utilizzati per pianificare nuovi aggiornamenti.

DevOps deriva in parte da Agile, di cui successivamente elencheremo i principi contenuti nel suo manifesto, poiché anche con DevOps si dà precedenza al “fare” e ai rilasci veloci. Un altro aspetto derivato da Agile è la documentazione, che viene ridotta al minimo, ma che comunque ha una certa importanza per le modifiche future [5].

1.2. Metodo Agile

Agile è una metodologia di sviluppo software basata su un insieme definito di principi derivanti direttamente o indirettamente dal “Manifesto Agile”, pubblicato nel 2001 da Kent Beck, Robert C. Martin ed altri [6]. I metodi “agili” si contrappongono ai tradizionali modelli di sviluppo e hanno come obiettivi principali la piena soddisfazione del cliente, l'abbattimento dei costi e dei tempi di sviluppo del software e l'aumento della qualità di quest'ultimo.

I principi dichiarati dal “Manifesto Agile” [7] sono i seguenti:

- Dare priorità massima al cliente.
- Accogliere i cambiamenti nei requisiti, anche a stadi avanzati dello sviluppo.
- Consegnare frequente di software funzionante, in maniera continua.
- Committenti e sviluppatori devono lavorare insieme per tutta la durata del progetto.
- Fondare i progetti su individui motivati, fornendo loro l'ambiente e tutto il supporto di cui hanno bisogno.
- Comunicare faccia a faccia all'interno del team in quanto risulta il metodo di comunicazione più efficiente.
- Il progresso è misurato in quantità di software funzionante rilasciato.
- Tenere continua attenzione all'eccellenza tecnica e alla buona progettazione.

- Dare priorità alla semplicità, per ridurre la quantità di lavoro svolto e realizzare prodotti semplici ed efficaci.
- La progettazione migliore deriva da team che si auto-organizzano.
- Ad intervalli regolari il team riflette su come aumentare la propria efficacia, dopodiché adatta il proprio comportamento di conseguenza.

I benefici acquisiti dall'azienda nell'adottare la metodologia Agile forniscono a quest'ultima maggiore flessibilità, produttività e trasparenza, oltre ad una qualità superiore del prodotto e un maggior coinvolgimento di tutte le parti interessate.

Successivamente alla definizione del manifesto Agile, da cui deriva DevOps, si passa ad una descrizione dei vantaggi apportati da questa metodologia di sviluppo.

1.3. I vantaggi di DevOps

Secondo Amazon, adottare DevOps apporta a svariati vantaggi, positivi per l'azienda che decide di utilizzare questa metodologia di sviluppo [4]:

Velocità: Promuove il rilascio di aggiornamenti software con maggiore frequenza grazie all'utilizzo di Microservizi e Continuous Delivery. La velocità apportata da DevOps permette all'azienda di essere reattiva nei confronti delle richieste di mercato e di ottenere rapporti con un maggior numero di clienti contemporaneamente.

Distribuzione rapida: Velocizza il rilascio di nuove versioni con bug fix e funzionalità aggiuntive, permettendo al cliente di iniziare fin da subito ad utilizzare le principali funzionalità del sistema, le quali risulteranno più stabili. Utilizza la Continuous Integration e Continuous Delivery.

Affidabilità: Verifica che gli aggiornamenti siano sempre conformi agli standard di qualità tramite l'integrazione continua e la distribuzione continua. Monitoraggio e registrazione di log consentono di mantenere sotto

controllo le prestazioni del sistema in tempo reale per far comprendere all'azienda l'impatto che ha avuto l'aggiornamento sugli utenti.

Scalabilità: L'automazione consente di gestire sistemi complessi e soggetti a variazioni di scala nel tempo. La scalabilità è alla base dei sistemi cloud in quanto permette al sistema di espandersi o restringersi in base al carico di utenza a cui viene posto.

Collaborazione: migliorata tra i team grazie alla fusione tra team di sviluppo e di produzione. La fusione garantisce una migliore comunicazione e collaborazione, migliorando anche la qualità del prodotto rilasciato.

Sicurezza: Il modello DevOps utilizza policy automatizzate per garantire la sicurezza anche grazie alle Policy come codice che utilizza il cloud per la definizione delle conformità che può essere monitorata e riconfigurata in modo automatico. Le risorse non conformi verranno segnalate in modo automatico ai team che potranno controllare ulteriormente.

Successivamente alla descrizione dei vantaggi apportati da questa metodologia si passa ad una descrizione delle tecnologie che abilitano quest'ultima all'interno delle aziende che decidono di adottarla.

1.4.Tecnologie

Verranno introdotte adesso le tecnologie utilizzate principalmente con DevOps. Inizialmente si parlerà di Jenkins, un orchestratore del ciclo di vita di un'applicazione sviluppata con DevOps. Successivamente verrà descritto BlueOcean, utilizzato per migliorare l'interfaccia di Jenkins e per avere una migliore user-experience. In ultimo verranno descritti Git e GitHub, utilizzati dagli sviluppatori per gestire le differenti versioni del codice.

1.4.1. Jenkins

Jenkins è un tool open source che ha come obiettivo principale quello di automatizzare alcune parti dello sviluppo software favorendo la Continuous Integration e il Continuous Delivery. Jenkins è scritto in java ed era un progetto inizialmente sviluppato da Oracle, ma successivamente, sviluppato da Hudson [8].

All'interno di Jenkins vi è l'utilizzo di diversi concetti quali:

Pipeline: sequenza automatizzata di passaggi che ogni aggiornamento software attraversa prima di essere rilasciato. Gli step principali sono build, test e release, ma è possibile aggiungerne altri liberamente, anche allo scopo di integrare ulteriori tecnologie all'interno della pipeline. Se uno step fallisce l'intera pipeline si blocca per quell'aggiornamento, sarà compito dello sviluppatore la risoluzione delle problematiche riscontrate e il riavvio della pipeline.

Plugins: componenti aggiuntivi del sistema che permettono l'aggiunta di funzionalità a Jenkins. I plugin vengono utilizzati anche per permettere l'utilizzo di tool esterni con Jenkins (es. Git, Maven ecc.). Esistono più di 1000 plugin e c'è la possibilità di crearne uno nuovo all'occorrenza.

Job: rappresenta un progetto su Jenkins.

L'architettura di Jenkins è distribuita ed è basata su due componenti [9]. Il primo di questi è Jenkins server, il quale è una dashboard web disponibile di default sulla porta 8080. Di default un nodo è già configurato nel Jenkins server e viene utilizzato come nodo master. Il ruolo del master è quello di programmare i job, distribuire le build agli slave, monitorare gli slave e presentare i risultati delle build agli sviluppatori. Tutto ciò è disponibile allo sviluppatore o all'amministratore del sistema tramite la sua interfaccia web da cui è possibile accedere per programmare le azioni che il nodo effettuerà in modo totalmente automatico. È possibile aggiungere altri nodi slave specificando l'indirizzo IP, lo username, la password e il protocollo di comunicazione del nodo sulla dashboard di Jenkins nel nodo master.

Il secondo componente dell'architettura distribuita di Jenkins è il node/slave/build server. Il ruolo di ogni Jenkins node dipende da ciò che viene definito nel Jenkins server, il che include l'esecuzione dei job con le build ricevute dal master, l'esecuzione dei test automatici sul prodotto, la delivery verso il cliente dell'applicazione sviluppata e testata. È possibile configurare un job in modo tale che questo possa essere eseguito solo su un nodo, solo su un particolare tipo di nodo o su di un qualsiasi nodo libero in quel momento. Un lato vantaggioso è che un nodo slave può essere una macchina configurata con qualsiasi sistema operativo.

Un classico flusso di azioni per effettuare la Continuous Integration con Jenkins consiste in diversi passi.

Prima di tutto viene effettuata la scrittura del codice su un IDE e il suo invio su di un repository di GIT. Il server di Jenkins controlla ad intervalli di tempo regolari il repository per verificare la presenza di aggiornamenti. Quando viene effettuato un commit da parte di uno sviluppatore, Jenkins rileva l'aggiornamento del codice nel repository, preleva quest'ultimo e prepara una nuova Build. A questo punto si possono verificare due eventi diversi. Se la build fallisce Jenkins invia una notifica al team di sviluppo che dovrà provvedere a verificare e risolvere la problematica riscontrata. Se invece la build ha successo, Jenkins effettua il deploy del codice su un server di test.

Il server effettua sei test unitari e test di integrazione automatici e restituisce a Jenkins il report sui risultati. Successivamente alla ricezione dei report, Jenkins genera un feedback per gli sviluppatori contenente i risultati dei test. Anche in questo caso si possono verificare due casi diversi: In caso di fallimento dei test, gli sviluppatori dovranno risolvere i bug rilevati. Successivamente alla correzione degli errori riscontrati viene effettuato un'ulteriore commit, Jenkins intercetterà questo evento e farà ripartire il processo automatizzato di build e test. In caso di successo dei test il codice viene poi rilasciato, per permettere l'installazione del prodotto sul server del cliente.

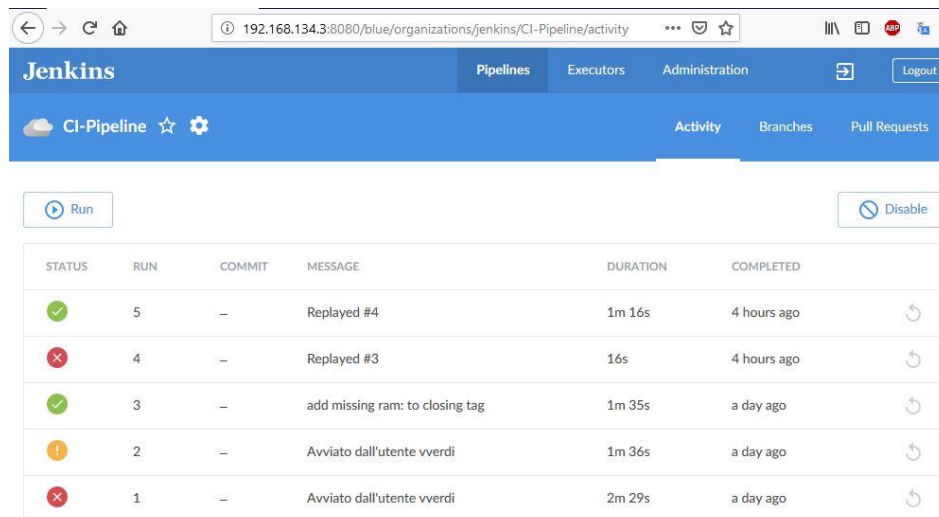
I vantaggi di Jenkins sono molteplici, ne elenchiamo qualcuno:

- Open source, user-friendly e facile da installare.
- È gratis, dunque preferibile dalle aziende che vogliono ridurre i costi per il mantenimento dei software aziendali.
- È facilmente modificabile ed estensibile tramite plugin. Effettua il deploy automatico del codice sul server di test e genera dei report con i risultati visibili agli sviluppatori.
- Disponibile per qualsiasi piattaforma.
- Disponibilità di diversi plugin creati dalla community per ampliare e automatizzare il sistema, aggiungendo ulteriori funzionalità all'occorrenza.
- Possibilità degli sviluppatori di scrivere dei test che verranno eseguiti in automatico da Jenkins.
- I problemi vengono scoperti e risolti velocemente grazie all'elevata automazione, il che permette il rilascio di codice qualitativo con altrettanta velocità.
- L'integrazione del codice viene effettuata in maniera automatica, così da ottimizzare i tempi e dunque i costi del progetto.

1.4.2. Blue Ocean

Blue Ocean rinnova l'interfaccia utente di Jenkins aggiungendo diverse funzionalità tra cui:

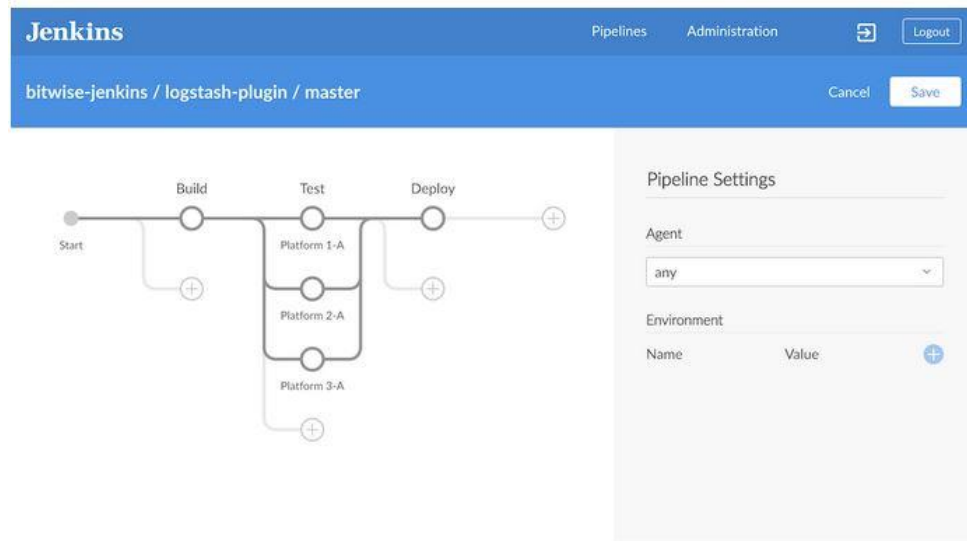
Interfaccia utente migliorata, permette di visualizzare la Continuous delivery pipelines, rendendo l'esperienza più intuitiva.



STATUS	RUN	COMMIT	MESSAGE	DURATION	COMPLETED
✓	5	—	Replayed #4	1m 16s	4 hours ago
✗	4	—	Replayed #3	16s	4 hours ago
✓	3	—	add missing ram: to closing tag	1m 35s	a day ago
!	2	—	Avviato dall'utente vverdi	1m 36s	a day ago
✗	1	—	Avviato dall'utente vverdi	2m 29s	a day ago

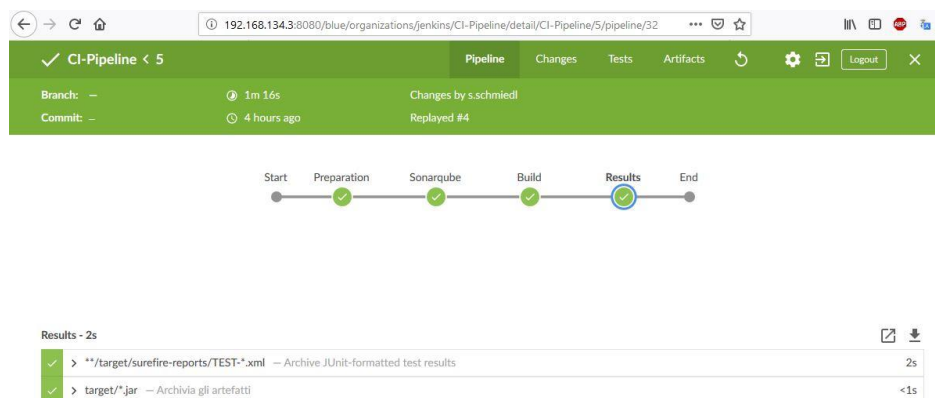
Nell'immagine sovrastante viene visualizzata la serie di compilazioni effettuate per la pipeline “CI-Pipeline”, ognuna avente diverse informazioni quali: stato finale, numero della compilazione, messaggio restituito, durata della compilazione e orario di conclusione. L'interfaccia utente di Blue Ocean facilita lo sviluppatore nei suoi compiti attraverso l'intuitività e la semplificazione di quest'ultima.

Pipeline editor, permette la creazione di una pipeline guidando l'utente con un'apposita interfaccia. Permette di programmare anche i task da parallelizzare, come è possibile vedere nella seguente immagine. I task parallelizzati compaiono come nodi sovrastanti aventi inizio e fine in comune. Questo tipo di task permette di ottimizzare l'esecuzione della pipeline risparmiando tempo e migliorando le prestazioni.



Precisione millimetrica, permette di localizzare con precisione dov'è situato l'errore scatenato durante la compilazione mostrando l'eccezione lanciata all'interno dell'interfaccia utente.

Nell'immagine seguente è possibile vedere l'interfaccia proposta da Blue Ocean, in questo caso la pipeline ha concluso il suo compito con successo e dunque il colore prevalente è il verde. Nel caso di fallimento la parte superiore dell'interfaccia verrebbe visualizzata di colore rosso come anche lo stage in cui si ha avuto il fallimento.



1.4.3. Git e GitHub

Git è un sistema di controllo delle versioni che permette di tracciare i cambiamenti effettuati sui file e di coordinare le attività di più persone che lavorano sugli stessi file. Fu sviluppato da Linus Torvalds nel 2005 per supportare l'attività di sviluppo del kernel Linux [10]. A differenza di altri sistemi di controllo delle versioni, tutta la cronologia dei cambiamenti non è in unico posto ma la copia di ogni sviluppatore costituisce un repository che contiene la storia di tutti i cambiamenti.

Spesso Git viene utilizzato in simbiosi con GitHub che è un servizio di hosting web. è stato sviluppato a partire dal 2008 Chris Wanstrath, PJ Hyett e Tom Preston-Werner e rilasciato per la prima volta nel 2009 [11]. Sebbene sia utilizzato principalmente per supportare lo sviluppo software, permette di salvare anche documenti di testo, fogli di lavoro Excel, foto e altro ancora. Su GitHub si possono trovare moltissimi repository pubbliche di cui è possibile effettuarne il clone. Clonare un repository vuol dire sostanzialmente copiarne l'intero contenuto sul nostro client. In questo modo si può consultare il codice più comodamente e magari provare ad eseguirlo. Eventuali modifiche restano soltanto locali e non ne viene tenuta traccia in alcun modo. Se una volta clonato e modificato un nostro progetto vogliamo convalidare i cambiamenti, prima di tutto bisogna effettuare un commit. Un commit indica che qualsiasi modifica è stata confermata. è buona pratica dunque effettuare commit frequenti e commentarli bene in modo tale da rendere più chiaro possibile i cambiamenti fatti. In ogni caso anche se si è effettuato il commit, le modifiche sono ancora locali e nessun altro può vedere le variazioni. Se vogliamo che i cambiamenti siano apportati anche sul server e che quindi sia premesso a tutti di visualizzarli, è necessario effettuare un push. Come è stato già anticipato, Git è pensato per supportare team di sviluppatori e quando si lavora ad un progetto spesso si ha bisogno di restare aggiornati sulle modifiche apportate dagli altri. A questo scopo esiste il pull che permette di scaricare la versione più aggiornata per poterci poi lavorare in locale. Se un membro del team deve occuparsi dello sviluppo di una particolare funzionalità senza andare a bloccare il lavoro degli altri può

decidere di creare un branch, cioè andare a creare una ramificazione del progetto. In questo modo i push effettuati sul branch non andranno ad intaccare il ramo originario. Una volta arrivati al completamento del lavoro per il quale era stato creato il branch, però, si vorrebbe poter apportare le modifiche anche sul ramo principale del progetto. Per fare questo basta semplicemente effettuare un merge. Eseguire il fork

di un repository è molto diverso dal crearne un branch. Prima di tutto va specificato che il forking non è una funzionalità di Git come quelle elencate sopra, ma è una caratteristica messa a disposizione da GitHub [12]. Come per il branch viene eseguita una copia della repository originaria ma questa volta viene creato un progetto diverso. Tutti i push, i branch e in generale i cambiamenti apportati alla nuova repository non andranno ad influenzare quella originale.

Capitolo 2: Problemi di security in DevOps

Come ogni altra metodologia di sviluppo software anche DevOps soffre di diverse problematiche di sicurezza. In questo capitolo se ne elencheranno alcune delle più note. Verranno descritte diverse tipologie di vulnerabilità del software e di come queste vengano catalogate, descrivendone le classificazioni e le differenze tra le varie metodologie. Infine, verrà trattato il threat modeling, descrivendone tutte le tipologie e alcune differenze tra loro.

2.1. Problematiche della metodologia

Verranno elencate adesso le principali problematiche di cui DevOps soffre [13]:

- Considerare la security come un problema secondario rispetto ad elementi come la rapidità di sviluppo e il rilascio dei servizi.
- Considerare la security come un qualcosa che possa rallentare il processo produttivo e il rilascio continuo degli aggiornamenti.
- La propagazione rapida di bug o problemi di sicurezza dal loro sviluppo alla distribuzione, causata dalla velocità apportata da DevOps.
- La scarsa protezione dei dati che si spostano tra più server, in base allo stadio della pipeline di sviluppo, testing e distribuzione del codice.
- L'aggiornamento continuo di software di terze parti, inclusi nel sistema, che senza i dovuti controlli può comportare l'introduzione di codice malevolo, il quale verrà anch'esso diffuso velocemente tra macchine del sistema sino al cliente, che a sua volta potrebbe avere anche molteplici utenti connessi al sistema acquistato.
- L'assenza di personale qualificato in sicurezza che effettui dei test mirati e supporti il team di sviluppo in tutte le fasi affrontate rappresenta un male per l'intera azienda.
- Tralasciare completamente o in parte l'analisi delle vulnerabilità o la sua esecuzione soltanto alla fine del processo produttivo espone il sistema a

molteplici rischi. Nell'ultimo caso essendo eseguita soltanto alla fine rende più complessa e lunga la risoluzione dei problemi riscontrati.

- La mancanza di una marcata separazione dei ruoli all'interno dei team, il che permette a chiunque abbia accesso al sistema di effettuare un'operazione qualsiasi.
- La mancanza di una sicurezza consolidata dell'ambiente di sviluppo

Dopo aver messo in chiaro quelle che sono le problematiche della metodologia DevOps, chiariamo, in linea di massima, quali sono le vulnerabilità del software, per permettere al lettore di essere a conoscenza delle diverse controversie che possono verificarsi dopo il rilascio di un software non sicuro.

2.2. Vulnerabilità del software

Le vulnerabilità del software rappresentano delle debolezze che possono essere sfruttate dall'attaccante per perseguire scopi malevoli sul sistema attaccato. Esistono diverse vulnerabilità del software, elenchiamo le vulnerabilità più note, descrivendole e mostrando azioni possibili per evitare di introdurre, durante lo sviluppo di un sistema, vulnerabilità di questo tipo.

Software da fonte non fidata: L'esecuzione di software proveniente da una fonte non fidata è una grave minaccia per il sistema. Questa vulnerabilità è causata da utenti inesperti e non dal sistema stesso. Le contromisure adottate per arginare il problema sono la formazione del personale e le soluzioni tecniche per impedire l'esecuzione del software in questione.

Incorrettezza del software: un programma è corretto quando su qualsiasi input che soddisfa le precondizioni l'output soddisfa le post-condizioni. Programmi non corretti costituiscono una minaccia poiché hanno comportamenti inattesi. Un programma corretto è vulnerabile se esiste un input che non soddisfa le precondizioni per cui non c'è un controllo e una gestione dell'errore adeguata.

Input del software da sorgente non fidata: l'input di un programma può provenire da una sorgente non fidata, in questo caso il programma se vulnerabile costituisce una minaccia poiché la sorgente può sfruttare la vulnerabilità per effettuare operazioni malevole. L'input deve essere validato, altrimenti il comportamento del programma potrebbe essere imprevedibile.

Code Injection: causato dalla mancanza di un corretto controllo sui dati in input, il code injection consiste nell'inserimento di codice malevolo tramite l'input fornito al programma.

SQL Injection: Tecnica di code injection usata per attaccare applicazioni per la gestione di dati. La tecnica prevede l'inserimento di input particolare al programma il quale esegue delle query aventi come parametri ciò che è stato inserito e restituisce un output diverso da ciò che il programmatore ha pianificato. Attacchi di questo tipo sono più semplici da effettuare se si conoscono il nome delle tabelle e colonne o su sistemi open source.

XSS (Cross Site Scripting): Vulnerabilità di siti web dinamici che non controllano approfonditamente gli input delle form. La vulnerabilità permette all'attaccante di eseguire script malevoli. Può essere di diversi tipi:

- *Non-Persistent XSS:* Possono essere eseguiti iniettando del codice html contenente script insieme ad un link che l'utente clicca. L'utente vedrà solo la scritta del link e non il codice con lo script. Una volta che l'utente clicca sul link, il browser eseguirà lo script inviando, come da codice, il proprio cookie contenente i dati della sua sessione all'attaccante il quale potrà utilizzarli per accedere al sito autenticandosi come l'utente attaccato. Per evitare questo tipo di attacco si potrebbero adottare alcune strategie, elencandone qualcuna:
 - Analizzare l'input per eliminare eventuali codici presenti.
 - Rilevare gli accessi simultanei e invalidare le sessioni.

- Richiedere di inserire di nuovo la password attuale prima di poterla cambiare (se l'attaccante decidesse di cambiare la password alla vittima dopo l'accesso)
- *Persistent XSS*: Possono essere eseguiti facendo eseguire degli script dal browser della vittima senza che quest'ultima se ne accorga. La pratica sta nello sfruttare la vulnerabilità di un sito per inserire in un commento (ad esempio) del codice html contenente un tag script che verrà eseguito quando la vittima caricherà i commenti. Lo script ad esempio può inviare all'attaccante i cookie della propria sessione e quest'ultimo potrà utilizzarli per autenticarsi sotto il nome della vittima. Per difendersi da questo tipo di attacchi è possibile disattivare gli script dal browser, oppure il sito potrebbe analizzare i commenti prima di pubblicarli per eliminare eventuali script.
- *DOM-based XSS*: Una vulnerabilità di questo tipo permette all'attaccante di iniettare codice JavaScript all'interno del sito. Il codice iniettato permette di riflettere i dati inseriti in quella pagina in un'altra pagina, il tutto client-side, così da non toccare il server per ridurre i rischi di essere scoperti. Un modo per difendersi da questo tipo di attacchi è disattivare gli script client-side dal browser.

CSRF (Cross-Site Request Forgery): Vulnerabilità causata dal mancato controllo dell'intenzionalità di una richiesta http da parte di un utente. La vittima invia involontariamente una richiesta HTTP dal suo browser al sito web dove è autenticato. Il sistema esegue la richiesta senza accertarsi del fatto che l'utente voglia davvero compiere quell'azione (ad esempio un bonifico dal proprio conto). L'attacco può cominciare da click dell'utente su un link o direttamente attraverso il caricamento di un'immagine da parte del browser web.

Ognuna di queste vulnerabilità note rappresenta un problema più o meno grave all'interno di un sistema informatico. Se all'interno di un sistema utilizzato da

molti si viene a conoscenza dell'esistenza di una vulnerabilità software occorre che questa venga catalogata e divulgata attraverso i canali informativi ufficiali affinché possa raggiungere sia gli sviluppatori del sistema, che dovranno rimediare alla problematica al più presto, sia agli utenti del sistema, che dovranno porre maggiore attenzione durante il suo utilizzo. Si tratterà ora della catalogazione delle vulnerabilità software.

2.3.Catalogazione delle vulnerabilità

La catalogazione di una vulnerabilità è un'operazione effettuata per determinarne la gravità, derivata dal tipo di vulnerabilità e alle risorse del sistema coinvolte, e per diffondere attraverso canali ufficiali la sua esistenza, associando anche una possibile mitigazione, per evitarne la diffusione. Esistono diversi metodi per catalogare vulnerabilità elencheremo successivamente i più noti:

Common Vulnerability Exposures (CVE).

Le vulnerabilità inserite nel CVE violano almeno una proprietà tra: confidenzialità, integrità e disponibilità. Le vulnerabilità inserite sono identificate da una stringa univoca CVE-ANNO-NUMERO e sono descritte da una scheda contenente descrizione, URL a una pagina dettagliata e data di creazione. CVE enumera le vulnerabilità ma non misura l'impatto e non indica quale tra le vulnerabilità debba essere gestita più urgentemente.

Common Vulnerability Scoring System (CVSS).

Stima la gravità di ogni vulnerabilità del catalogo CVE dandone una valutazione numerica da 0 a 10. CVSS assegna il punteggio in base a tre gruppi di metriche:

- Base metric: stimano la gravità della vulnerabilità a prescindere da fattori temporali e ambientali
- Temporal metric: stimano la gravità della vulnerabilità dal punto di vista temporale

- Environmental metric: stimano la gravità dal punto di vista ambientale.

Ad ogni metrica è associata una domanda a risposta multipla, ciascuna risposta fornisce un peso numerico. I singoli valori ottenuti dalle diverse metriche vengono poi aggregati mediante una serie di formule.

Common Weaknesses and Exposures (CWE).

Il catalogo CWE è un insieme di oggetti, ciascuno con identificatore e attributi. Gli oggetti del catalogo CWE possono essere legati da una relazione padre-figlio. Oggetti di tipo Category puntano ad altri che condividono uno specifico attributo, un oggetto Compound mette in relazione diverse debolezze correlandole se, sfruttandole insieme, provocano una vulnerabilità (composite object). Un oggetto Compound può anche aggregare tutte le debolezze che sfruttate in cascata provocano una vulnerabilità (chain object). L'oggetto di tipo View punta a una serie di oggetti che condividono un certo attributo, può essere di tipo graph o di tipo slice.

Common Weakness Scoring System (CWSS).

È simile al CVSS, con la differenza che il punteggio assegnato ad ogni CWE è da 0 a 100 e che CWSS può essere utilizzata prima che la vulnerabilità sia scoperta e verificata. Il punteggio è dato dal prodotto di tre fattori: Base finding, che stima la robustezza dei meccanismi di protezione, l'Attack surface, stima le barriere che un attaccante deve superare per sfruttare la debolezza e l'Environmental che stima l'impatto che ha la debolezza sul sistema.

Si fa presente che prima di catalogare una entità dovranno essere ben chiare le sue origini e la sua gravità. Un valido metodo per avere una vista dall'alto del sistema utile per scovare l'origine e la gravità delle sue eventuali vulnerabilità è il Threat Modeling.

2.4. Threat modeling

Molti software al giorno d'oggi devono far fronte a diverse minacce (threats) e il numero di minacce cresce al variare delle tecnologie. Una minaccia rappresenta una potenziale violazione della security del sistema. Le minacce possono provenire dall'esterno o dall'interno delle organizzazioni e possono puntare a disabilitare i sistemi o al furto dei dati. Per prevenire le minacce si può utilizzare il Threat modeling, usato per creare un'astrazione del sistema, profilare gli attaccanti, compreso i loro obiettivi e metodi e fornire una catalogazione delle potenziali minacce [14].

Esistono molti threat model diversi, ne verrà elencato qualcuno spiegando le sue caratteristiche [15]:

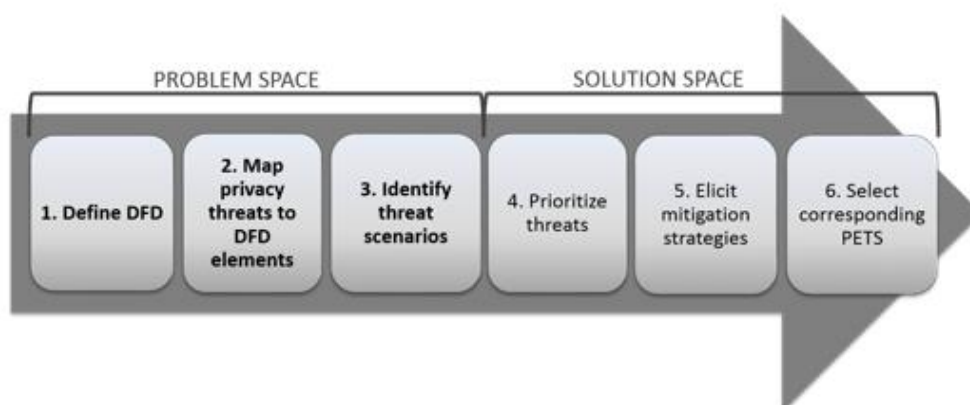
STRIDE: è attualmente il più maturo metodo di threat modeling siccome è nato nel 1999 e nel corso degli anni ha incluso nuove tabelle per le minacce. STRIDE è usato per identificare le entità del sistema, gli eventi e i limiti.

	Threat	Property Violated	Threat Definition
S	Spoofing identify	Authentication	Pretending to be something or someone other than yourself
T	Tampering with data	Integrity	Modifying something on disk, network, memory, or elsewhere
R	Repudiation	Non-repudiation	Claiming that you didn't do something or were not responsible; can be honest or false
I	Information disclosure	Confidentiality	Providing information to someone not authorized to access it
D	Denial of service	Availability	Exhausting resources needed to provide service
E	Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

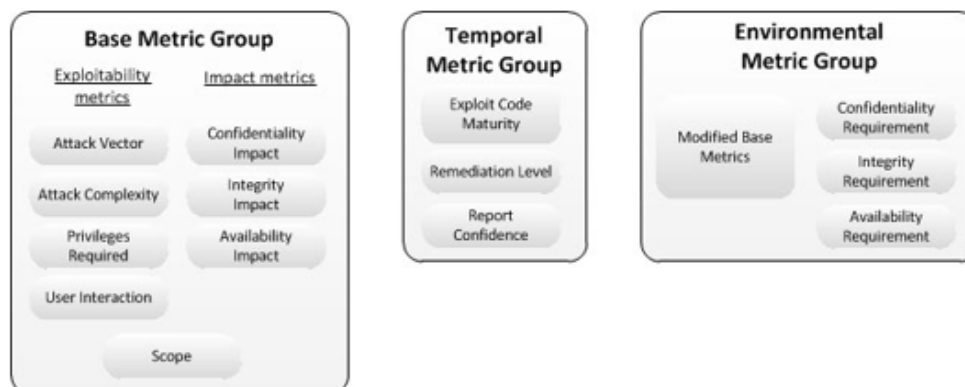
PASTA (Process for Attack Simulation and Threat Analysis): è un threat modeling formato da sette stadi, ognuno con diverse attività da svolgere, come mostra la figura. PASTA si basa sull'unione di obiettivi di business e dei requisiti tecnici.



LINDDUN (Linkability, Identifiability, Nonrepudiation, Detectability, Disclosure of information, Unawareness, Noncompliance): basato su concetti di privacy, è utilizzabile per la sicurezza dei dati. È formato da sei passi, come mostrato in figura:



CVSS: Una prima descrizione è già stata fatta precedentemente, CVSS è utilizzato in combinazione con altri metodi di threat modeling e attribuisce un punteggio ad un sistema sulla base di tre gruppi di metriche:



Attack Trees: È uno dei metodi più vecchi ed inizialmente veniva utilizzato da solo mentre adesso lo si utilizza in combinazione con altri metodi di threat modeling come ad esempio STRIDE, CVSS e PASTA. In caso di un sistema complesso è possibile creare un albero d'attacco per ogni componente. L'albero d'attacco può essere costruito anche dall'amministratore del sistema per migliorare la sicurezza

Persona non Grata (PnG): Metodo che si focalizza sulle abilità e sulle motivazioni degli attaccanti umani. Vengono scritte delle schede riguardanti ogni tipologia di attaccante, di fatti si abbina con l'approccio Agile che utilizza le personas. Una volta identificate le motivazioni e le abilità delle persone non grate ci si focalizza sulla ricerca delle vulnerabilità che queste persone potrebbero riscontrare nel nostro sistema

Security Cards: Non è un metodo formale, ma permette attraverso tecniche di brainstorming di ideare un deck di carte composto da possibili attaccanti. Il deck è composto da 42 carte di cui: 9 riguardanti l'impatto umano, 14 riguardo le motivazioni dell'attaccante, 11 sulle risorse dell'attaccante e 9 sui metodi utilizzati dall'attaccante

Human Impact	Adversary's Motivations	Adversary's Resources	Adversary's Methods
<ul style="list-style-type: none"> the biosphere emotional well-being financial well-being personal data physical well-being relationships societal well-being unusual impacts 	<ul style="list-style-type: none"> access or convenience curiosity or boredom desire or obsession diplomacy or warfare malice or revenge money politics protection religion self-promotion world view unusual motivations 	<ul style="list-style-type: none"> expertise a future world impunity inside capabilities inside knowledge money power and influence time tools unusual resources 	<ul style="list-style-type: none"> attack cover-up indirect attack manipulation or coercion multi-phase attack physical attack processes technological attack unusual methods

HTMM (Hybrid Threat Modeling Method): è un metodo formato dalla combinazione di altri metodi quali SQUARE (Security Quality Requirements Engineering Method), Security Cards e PnG. L'obiettivo del metodo ibrido è quello di non avere falsi positivi, threat trascurati e un risultato coerente indipendentemente da chi sta effettuando la modellazione dei threat. HTMM esegue i seguenti passi:

- Identificazione del sistema da modellare.
- Applicare Security Cards sui suggerimenti degli sviluppatori
- Rimuovere le persone non gradite (individuate mediante PnG)
- Continua con un metodo formale di valutazione dei rischi

Quantitative Threat Modeling Method: è un metodo ibrido formato da STRIDE, CVSS e albero d'attacco. Ha come obiettivo quello di affrontare la modellazione delle minacce per sistemi con complesse interdipendenze tra i diversi componenti. Il primo passo è quello di costruire l'albero d'attacco di ogni componente del sistema per le cinque categorie di minacce di STRIDE. Successivamente si applica CVSS per valutare la debolezza delle varie componenti dell'albero.

Trike: è stato creato come un framework per la sicurezza che utilizza la tecnica del threat modeling. Trike inizia con il definire il sistema costruendo un modello di requisiti composto da attori, asset, azioni e regole. Viene creata una matrice in cui le righe sono gli attori e le colonne sono gli asset. Ogni cella della matrice è divisa in quattro parti (una per ogni operazione CRUD). Ad

ogni cella è assegnato un valore tra: operazione abilitata, operazione disabilitata o azione regolata. Viene costruito un data flow diagram (DFD) dove ogni elemento è mappato ad una selezione di attori e asset. Iterando sul DFD si identificano i threat, ogni threat scoperto diventa la radice dell'albero d'attacco.

VAST (Visual, Agile and Simple Threat): è un metodo basato su Threat Modeler, una piattaforma di threat-modeling automatico. VAST richiede la creazione di due tipi di modelli. Il threat model delle applicazioni, mediante process-flow diagrams, rappresenta il punto di vista dell'architettura. Il threat model delle operazioni è creato dal punto di vista dell'attaccante ed è basato sul DFD (data flow diagram).

OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation): metodo basato sulla valutazione strategica dei rischi. OCTAVE è formato in tre fasi. Per prima cosa, costruisci i profili delle minacce basandosi sugli asset, successivamente identifica le infrastrutture vulnerabili ed infine sviluppa una strategia di sicurezza.

Capitolo 3: Mitigazione delle problematiche

Le problematiche relative all'utilizzo di DevOps rendono l'implementazione di questa metodologia particolarmente delicata nelle grandi aziende, dove l'introduzione di bug di sicurezza all'interno del prodotto può comportare a ingenti danni per l'azienda stessa o per il cliente qualora il software sviluppato sia indirizzato ad un dispositivo relativo alla salute dell'uomo, come quelli per l'IoT medicali o quelli per il controllo di centrali nucleari ecc.

In questo capitolo verranno definiti i principi su cui si basa la sicurezza informatica, passando poi alla definizione di una metodologia di sviluppo software in grado di mitigare gran parte delle problematiche di sicurezza riscontrate in DevOps: DevSecOps. Si parlerà infatti della sicurezza apportata dall'adozione di questa metodologia all'interno del ciclo DevOps.

3.1. Principi di sicurezza

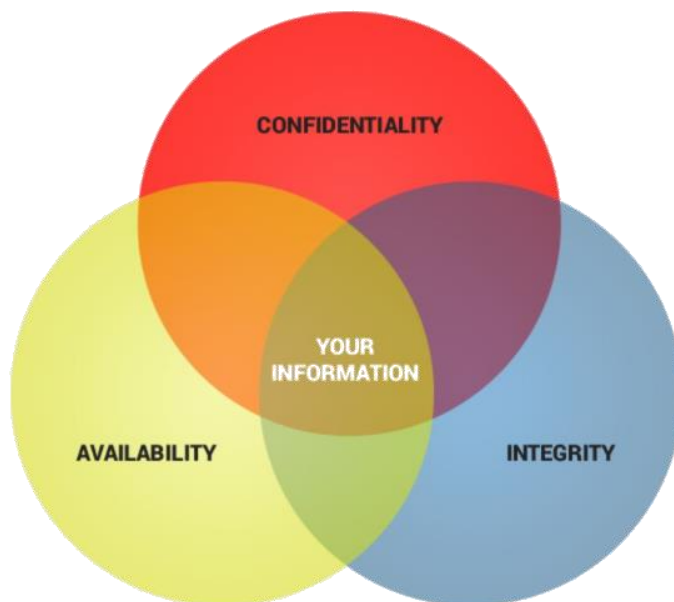
In informatica, il termine *sicurezza informatica* indica la protezione dell'informazione e dei sistemi informativi da accessi, utilizzi, modifiche, registrazioni o distruzioni non autorizzate. Lo scopo della sicurezza informatica è dunque quello di garantire tre punti fondamentali sono [16]:

Confidenzialità (confidentiality): Assicurare che le informazioni non siano accessibili a utenti non autorizzati. La confidenzialità deve essere assicurata lungo tutte le fasi di vita del dato, a partire dal suo immagazzinamento, durante il suo utilizzo o il suo transito lungo una rete di connessione. Vi sono svariati strumenti che possono essere utilizzati per garantire la confidenzialità delle informazioni: la crittazione delle comunicazioni, le procedure di autenticazione, la creazione di modelli di data governance ben definiti e le azioni di awareness sugli utenti.

Integrità (integrity): Assicurare che le informazioni non vengano alterate da persone non autorizzate.

L'integrità si divide in integrità dei dati e integrità dell'origine di questi ultimi. I meccanismi che assicurano l'integrità sono anch'essi suddivisi in due classi: meccanismi di prevenzione e meccanismi di rivelazione. I meccanismi di prevenzione mantengono l'integrità dei dati bloccando il permesso alla modifica dei dati a persone non autorizzate. I meccanismi di rivelazione riportano agli amministratori del sistema che l'integrità dei dati non è più affidabile.

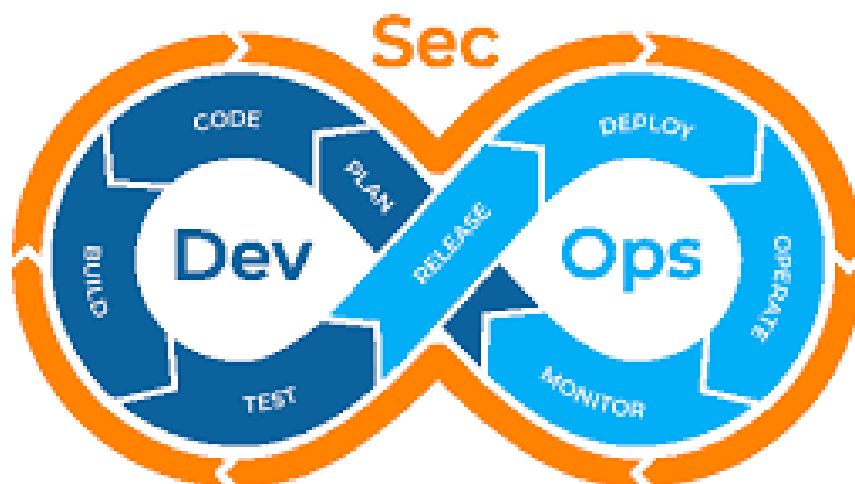
Disponibilità (availability): la disponibilità si riferisce alla possibilità, per i soggetti autorizzati, di poter accedere alle risorse di cui hanno bisogno per un tempo stabilito ed in modo ininterrotto. Rendere un servizio disponibile significa essenzialmente due cose: impedire che durante l'intervallo di tempo definito avvengano interruzioni di servizio e garantire che le risorse infrastrutturali siano pronte per la corretta erogazione di quanto richiesto [17].



3.2. DevSecOps

In questo paragrafo verrà definita la metodologia di sviluppo software DevSecOps, posta come una possibile soluzione alle problematiche di sicurezza insite nella metodologia DevOps. DevSecOps è sviluppata a partire da DevOps e ha come obiettivo principale la sicurezza informatica dello sviluppo software e delle applicazioni prodotte. Verrà successivamente descritta la sicurezza dell'ambiente e dei dati e la sicurezza del processo di Continuous integration e Continuous delivery apportata da questa nuova metodologia proposta.

3.2.1. Definizione



DevSecOps integra tutti i principi di DevOps, ma pone maggior riguardo alla sicurezza in ogni fase del ciclo di vita del software. In fase di sviluppo la sicurezza viene presa in considerazione mediante l'utilizzo di tool che supportano lo sviluppatore per evitare l'introduzione di falle di sicurezza durante la scrittura del codice, in fase di testing oltre a testare le funzionalità del sistema vengono inseriti anche dei test sulla sicurezza di quest'ultimo, test effettuati attaccando il sistema alla ricerca di falle di sicurezza e che permettono di avere dei risultati dettagliati che vengono poi recapitati allo

sviluppatore. Anche una volta rilasciato il sistema, durante la fase di monitoring, si tiene traccia, tra i diversi dati raccolti sull'utilizzo del sistema, anche di dati relativi alla sicurezza del prodotto per permettere un costante miglioramento su ogni aspetto. Infine, qualora lo si ritenga necessario, sulla base dei dati raccolti dai feedback forniti dall'applicazione e dagli utenti, è possibile pianificare eventuali rilasci contenenti modifiche all'applicazione per risolvere le eventuali problematiche riscontrate.

DevSecOps evidenzia la necessità di stimolare i team di sicurezza alla protezione delle informazioni e a definire un piano di automazione della sicurezza alla nascita delle iniziative DevOps, nonché di aiutare gli sviluppatori a creare codice tenendo a mente la sicurezza, un processo che richiede la condivisione di dati, feedback e informazioni sulle minacce note da parte dei team di sicurezza. Questo nuovo approccio potrebbe richiedere la partecipazione degli sviluppatori a nuovi corsi di formazione sulla sicurezza, cosa che non era così tipica con l'approccio di sviluppo tradizionale.

Sulla base della maggior attenzione sulla sicurezza apportata da DevSecOps verranno ora descritti i miglioramenti della sicurezza dell'intero ambiente e dei dati al suo interno.

3.2.2. Sicurezza dell'ambiente e dei dati

DevSecOps pone diverse regole per quanto riguarda la protezione di tutto l'ambiente DevOps e dei dati contenuti al suo interno [18].

La standardizzazione e l'automazione dell'ambiente è richiesta per far sì che ogni servizio limiti nella misura massima possibile gli accessi e i collegamenti non autorizzati.

La centralizzazione delle capacità di controllo degli accessi e delle identità degli utenti risulta fondamentale per garantire la sicurezza di applicazioni strutturate in microservizi. I meccanismi di autenticazione centralizzata e di controllo degli accessi devono inoltre essere molto rigidi poiché l'autenticazione avviene in più punti.

L'isolamento dei container e tra i container che eseguono microservizi dalla

rete è necessaria per garantire la sicurezza dei dati in transito e dei dati conservati poiché entrambi possono essere oggetto di attacchi che potrebbero provocare elevati danni all'azienda.

L'utilizzo di una piattaforma di orchestrazione dei container con funzione di sicurezza integrata e che cripta i dati scambiati tra app e servizi contribuisce a ridurre drasticamente le possibilità di accesso non autorizzato alle risorse.

L'introduzione di gateway di API protetti permette di accrescere la visibilità su autorizzazioni e Routing. Riducendo le API esposte è possibile ridurre la superficie d'attacco.

La sicurezza dell'ambiente e dei dati non è l'unica sicurezza apportata da DevSecOps. Verrà ora descritta la sicurezza del processo di CI/CD.

3.2.3. Sicurezza del processo CI/CD

DevSecOps richiede che vengano rispettati alcuni accorgimenti importanti per garantire un certo livello di sicurezza all'interno del processo di sviluppo, che applica le metodologie di Continuous integration e Continuous delivery [18].

L'integrazione di scanner di sicurezza che controllino i container dovrebbe rientrare nella fase di aggiunta di questi ultimi all'ambiente, per garantire che non vengano aggiunte anche falle di sicurezza o codice malevolo.

L'automatizzazione dei test di sicurezza nel processo di Continuous integration è fondamentale in quanto restituisce un doppio vantaggio poiché permette, nel momento in cui viene apportata una modifica ad una funzionalità e prima del suo rilascio, l'esecuzione totalmente automatica dei test e la restituzione dei report contenenti l'insieme di problematiche sicurezza riscontrate nel sistema. I report possono poi essere utilizzati per migliorare la sicurezza dell'applicazione sviluppata, effettuando delle azioni atte a mitigare le problematiche riscontrate.

In fase di accettazione delle modifiche, è necessario inserire dei test automatici per verificare la sicurezza riguardante gli input del sistema.

Implementare l'automatizzazione degli aggiornamenti di sicurezza, come le patch per le vulnerabilità note, direttamente nel flusso DevOps così da eliminare la necessità per gli amministratori di accedere ai sistemi di produzione ed effettuare l'operazione manualmente, tenendo un registro delle modifiche [19].

In conclusione, è possibile affermare che DevSecOps è la migliore alternativa per le aziende che intendono sviluppare in ambiente sicuro e veloce applicazioni altrettanto sicure che permettono uno stretto contatto con il cliente e un'elevata flessibilità in caso di cambio dei requisiti di progetto. DevSecOps permette, oltre a mantenere un buon livello di sicurezza all'interno del sistema sviluppato, di mantenere anche degli ottimi tempi di rilascio, grazie all'esecuzione dei test automatici di sicurezza che non rallentano elevatamente l'intero sistema. Conclusa questa parte introduttiva verrà ora descritta l'attività effettuata per applicare DevSecOps a partire da un sistema aziendale che utilizza DevOps, spiegando dettagliatamente sia il funzionamento di ogni software utilizzato, sia il suo utilizzo durante l'esperienza effettuata.

Capitolo 4: Sperimentare DevSecOps come soluzione ai problemi di sicurezza in una pipeline di CI/CD.

L'obiettivo della sperimentazione è quello di implementare la metodologia DevSecOps partendo dalla configurazione di un sistema che simuli un ambiente di sviluppo DevOps e successivamente verranno sviluppati al meglio gli aspetti riguardanti la sicurezza. Il sistema sarà formato da diverse macchine virtuali, ognuna di esse rappresenterà un nodo di Jenkins. Negli ambienti di sviluppo DevOps un nodo di Jenkins è un server che può essere situato in qualsiasi parte del mondo. Ogni nodo della rete avrà un determinato compito all'interno della pipeline di DevOps.

Per la creazione del sistema si parte dalla configurazione nodo master di Jenkins per poi creare altre macchine in base agli stage della pipeline che utilizzeremo. Alle macchine virtuali che compongono il sistema verrà aggiunta un'ulteriore macchina desktop che simulerà un probabile attaccante al sistema DevOps. Dopo aver mostrato le diverse tipologie di attacchi che si possono ricevere verrà mostrato l'utilizzo di software sulla macchina dell'attaccante per effettuare dei test di sicurezza, utilizzando gli eventuali risultati ottenuti dai test per migliorare la sicurezza di Jenkins. Questa fase si concluderà con la creazione del Threat Model del sistema generato.

Dopo aver effettuato i test di sicurezza e aver rilevato le problematiche insite nel sistema DevOps ci si focalizzerà sull'hardening, migliorandone così il più possibile tutti gli aspetti di sicurezza. Verranno infine sperimentati alcuni tool che permetteranno di effettuare test di sicurezza in modo automatico, inserendoli all'interno della pipeline di Jenkins, per aumentare quanto più possibile l'automatizzazione di tutto il processo produttivo e dell'analisi della sicurezza.

4.1. Test di sicurezza

Conclusa la fase di installazione e configurazione di Jenkins, verranno mostrate le esecuzioni di diversi test, effettuati mediante appositi software che attaccano il sistema alla ricerca delle problematiche di sicurezza esistenti. Per prima cosa verrà effettuata una descrizione dei possibili attacchi che si possono ricevere all'interno di un sistema per poi proseguire con la parte pratica del progetto.

Lo scopo di questa fase è di scovare le problematiche del sistema per poter poi mitigare quelle che eventualmente risulteranno più importanti. Per conseguire lo scopo è stata configurata una macchina virtuale avente come sistema operativo Ubuntu 18.04 LTS, versione desktop. Sulla macchina sono stati installati e configurati due software, i quali saranno prima introdotti spiegandone i diversi particolari e poi utilizzati per la ricerca di falle di sicurezza: OWASP ZAP e Nessus.

4.1.1. Tipologie di Attacchi

Un attacco informatico è un'azione malevola, compiuta da singoli individui o da intere organizzazioni, ai danni di sistemi informativi, sistemi di comunicazione, reti di calcolatori. Definiamo alcune tipologie di attacco.

Man in the middle: Una macchina si posiziona al centro della linea di conversazione tra due altre macchine. Se è di tipo passivo la terza macchina legge soltanto il flusso di dati scambiato, se è di tipo attivo la terza macchina si interpone fisicamente tra le altre due leggendo e modificando i pacchetti di dati scambiati. L'attacco di tipo attivo è più semplice su protocollo UDP mentre è più complesso su protocollo TCP.

Denial of Service (DoS): Attacco mirato a saturare le risorse dei calcolatori presenti sulla rete per interrompere il servizio. Soluzioni possibili a questo tipo di attacco sono: dimensionare dinamicamente la coda di backlog e diminuire il time to live per le richieste in attesa nella coda.

Distributed Denial of Service (DDoS): Attacco attuato mediante l'utilizzo di una Botnet (vedi malware per il controllo e l'attacco). Gli attaccanti non si espongono direttamente ma eseguono l'attacco mediante il controllo di diverse macchine infettate, dette zombie, le quali inondano i server di richieste di connessione saturando di fatto le loro risorse.

IP address spoofing: Lo spoofing è un attacco che consiste nella falsificazione dell'identità. L'IP spoofing consiste nel creare un pacchetto IP contenente un indirizzo falso del mittente. Un meccanismo per evitare l'IP spoofing è quello di evitare che da un'interfaccia di un router siano inviati pacchetti in cui l'IP sorgente non è quello che ci si aspetta. Un altro metodo consiste nell'utilizzo di tabelle di routing.

Session Hijacking: Si tratta di un furto dei cookies usati per autenticare un utente su un sistema remoto. I cookie vengono rubati tramite un calcolatore al centro della comunicazione o accedendo ai cookies della vittima. I metodi per evitare questo attacco sono: l'uso di un numero casuale molto lungo come chiave di sessione, per evitare che l'attaccante provi ad indovinarla, rigenerare l'ID della sessione dopo che si è effettuato il login, utilizzare la crittografia per i dati scambiati dalle due parti o aggiungere controlli per verificare che l'IP sia sempre lo stesso durante una sessione.

Clickjacking: attacco che consiste nel reindirizzare il click di un utente a sua insaputa su di un altro oggetto della pagina. Oltre all'intercettazione del click possono essere intercettati i tasti premuti da tastiera e dunque all'attaccante è possibile rubare password ed altre informazioni digitate. L'attacco viene effettuato tramite l'inserimento di codice JavaScript contenente un gestore dell'evento generato dal click, oppure inserendo una pagina trasparente sopra alla pagina reale (inner frame).

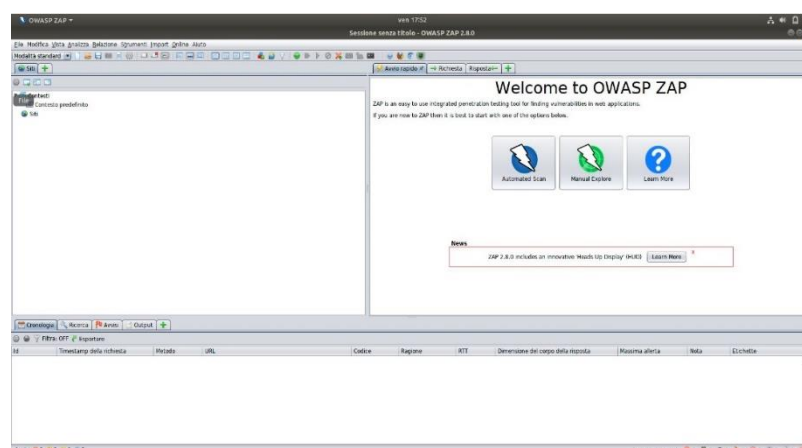
Dopo aver chiarito quelle che sono le varie tipologie di attacchi che si possono ricevere a danno di un sistema informatico, verranno descritti quali sono e come funzionano alcuni software di rivelazione delle vulnerabilità insite in un sistema. Per ogni software dopo una prima parte descrittiva verrà

mostrato come è stato utilizzato per rivelare le vulnerabilità esistenti nel sistema che applica la metodologia DevOps.

4.1.2. Testing con OWASP ZAP

OWASP Zed Attack Proxy (ZAP) è uno strumento di penetration testing che consente di rilevare vulnerabilità di applicazioni e siti web [20].

ZAP è composto da due sezioni principali: la prima è uno scanner automatizzato di vulnerabilità che consente di identificare i problemi e fornire report con dettagli delle vulnerabilità. La seconda sezione permette a ZAP di operare come proxy per ispezionare il traffico e tutta la comunicazione http e gli eventi, permettendo di modificarli o analizzare i loro trigger. La seconda modalità è manuale in quanto il traffico viene analizzato a partire dalle interazioni che l'utente effettua con il sito ispezionato mediante il browser.



Il tool è destinato a professionisti della sicurezza, sviluppatori, tester ed è disponibile per molti sistemi operativi aventi java installato.

Vi sono diverse modalità in cui ZAP può operare: Safe, Protected, Standard e Attack.

Safe impedisce ogni azione potenzialmente dannosa nei riguardi del sito target.

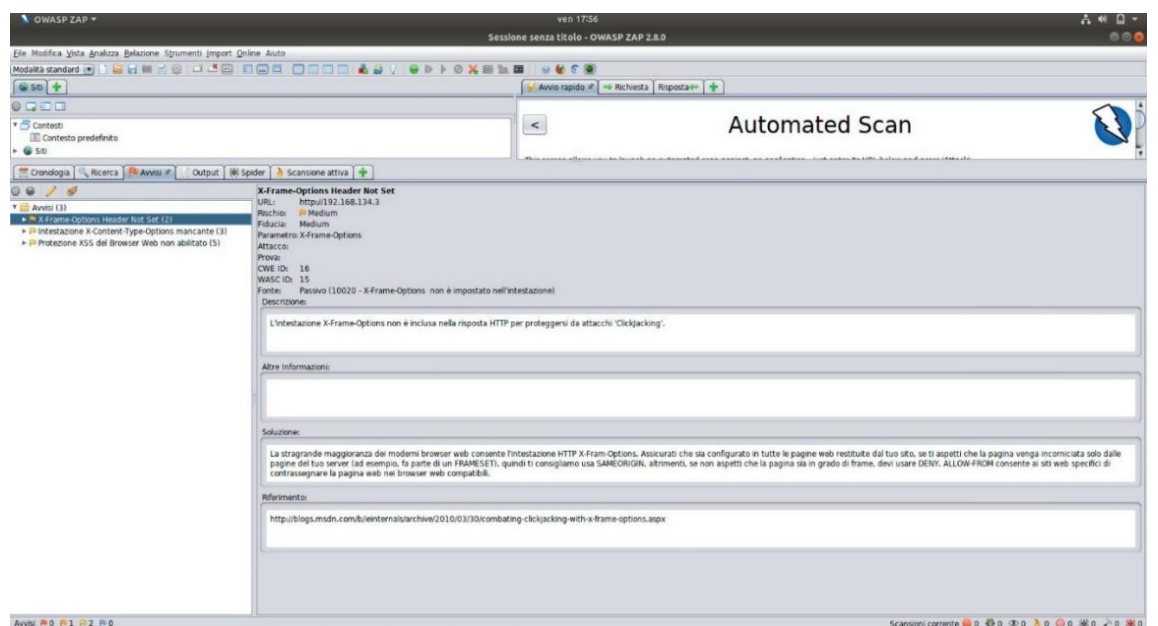
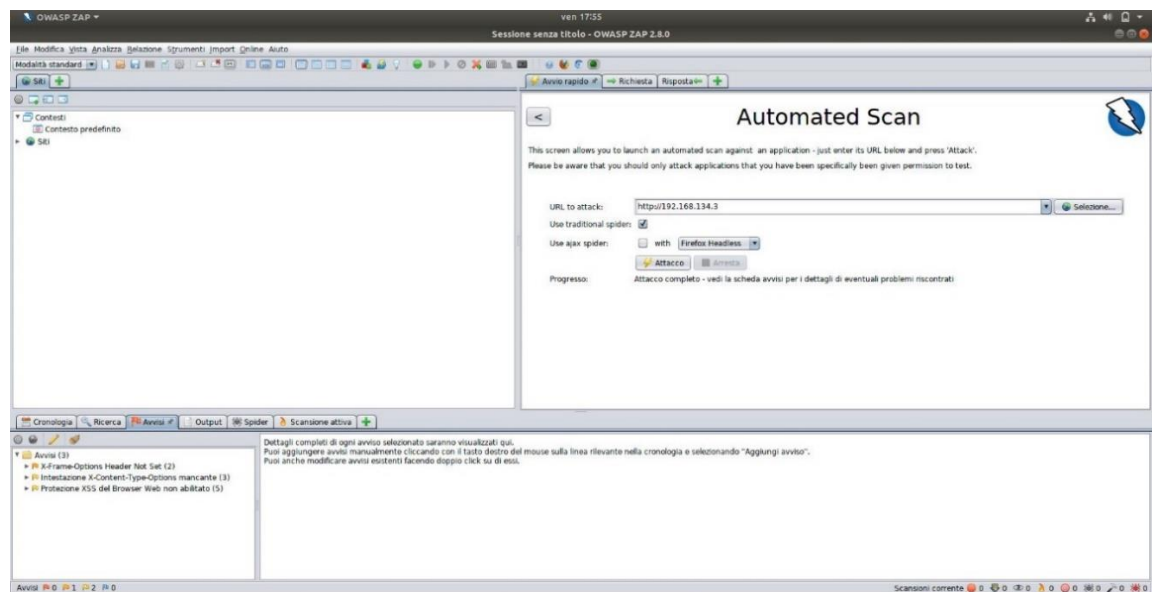
Protected permette azioni dannose negli URL definiti nello scope.

Standard è la modalità di default con cui si eseguono gli attacchi.

Attack esegue una scansione attiva su tutti i nodi nello scope non appena vengono scoperti e aggiunti.

I risultati vengono categorizzati con diversi colori, si parte dal rosso per quelli più gravi, poi arancione, giallo e infine azzurro. Cliccando sul problema riportato è possibile ottenere una possibile soluzione suggerita e un link alla documentazione.

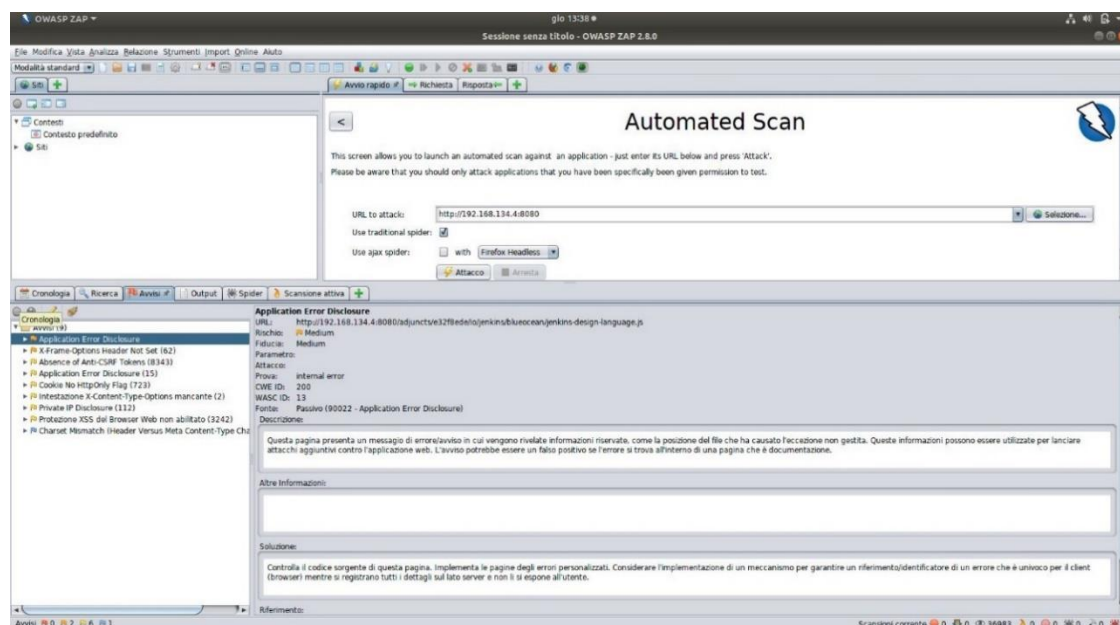
Una volta avviato OWASP ZAP sono stati eseguiti alcuni test automatici aventi come target l'IP della macchina su cui è installato Jenkins.



I test effettuati hanno rilevato diverse problematiche relative alla macchina su cui è in esecuzione Jenkins:

1. L'opzione X-Frame non è presente nell'header (Rischio medio):
L'opzione X-Frame dovrebbe essere inclusa nell'header per prevenire attacchi di tipo clickjacking, dove il click di un utente su di un oggetto viene reindirizzato a sua insaputa su di un altro oggetto.
La maggior parte dei browser permette di inserire questa opzione nell'header delle pagine restituite dal sito.
Per evitare che questa problematica sussista è necessario impostare X-Frame con valore SAMEORIGIN nell'header HTML.
2. L'intestazione X-Content-Type è mancante (Rischio basso):
Se X-Content-Type è impostato su "nosniff" non si permetterà al browser di sovrascrivere a propria discrezione i mime-type degli elementi inclusi nella pagina, evitando così possibili attacchi XSS.
Per eliminare questa problematica bisogna inserire questo header in tutte le pagine web.
3. L'intestazione X-XSS-Protection non è presente (Rischio basso):
L'header HTTP di risposta X-XSS-Protection è una funzionalità di diversi browser che impedisce alle pagine di caricarsi quando rilevano attacchi di tipo cross-site scripting reflected (XSS).
Per ovviare a questa problematica è necessario il campo X-XSS-Protection sia presente e impostato a '1' all'interno dell'header.

Successivamente sono stati effettuati dei test con OWASP ZAP mirati alla porta di Jenkins. I test hanno portato alla luce le seguenti problematiche di sicurezza relative a Jenkins:



1. Rivelazione di informazioni nel messaggio d'errore (Rischio medio):

Alcune pagine nel lanciare un messaggio di errore rivelano informazioni riservate, tra cui la posizione del file che ha lanciato l'eccezione non gestita.

La soluzione suggerita è di modificare queste pagine mostrando un identificativo per l'errore dato e mostrarlo soltanto sul server, nascondendolo all'utente.

2. Assenza di token anti-CSRF (Rischio basso):

Criticità rilevata in alcune form che permetterebbe un attacco di tipo Cross Site Request Forgery, è una vulnerabilità a cui sono esposti i siti web dinamici quando sono progettati per ricevere richieste da un client senza meccanismi per controllare se la richiesta sia stata inviata intenzionalmente oppure no. Il CSRF sfrutta la fiducia di un sito nel browser di un utente.

Le soluzioni a questa vulnerabilità sono diverse:

Non utilizzare il metodo GET per richieste che comportano un cambiamento di stato.

Usare framework, librerie per evitare l'introduzione della vulnerabilità.

Inviare una richiesta di conferma addizionale all'utente nel caso di operazioni pericolose.

Eseguire il logout dal sito prima di visitare altre pagine web.

3. Cookie senza flag HttpOnly (Rischio basso):

Un cookie è stato impostato senza il flag HttpOnly, è possibile dunque che venga inserito un cookie tramite JavaScript.

Per evitare questa problematica bisogna assicurarsi che il flag HttpOnly sia impostato per tutti i cookies.

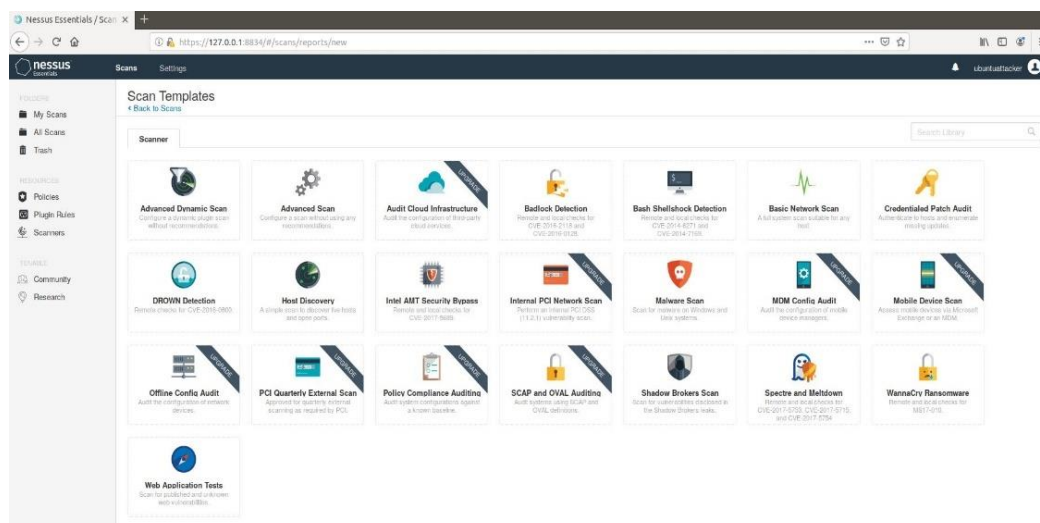
4.1.3. Test con Nessus

Nessus è un software proprietario progettato per rilevare vulnerabilità di tutti i tipi.

Il software è di tipo client-server ed è costituito da nessusd, il demone sul server che effettua la scansione e da Nessus, il client con interfaccia grafica.

Il client fornisce all'utente i risultati della scansione ed altri servizi quali l'abilitazione di diversi plugin, disponibili per il download e la configurazione, la possibilità di creare un nuovo plugin, la presenza di suggerimenti per la soluzione alle vulnerabilità riscontrate e riportate all'utente [21].

I tipi di test effettuati da Nessus vengono scelti in fase iniziale dall'utente ed alcuni sono già disponibili dopo l'installazione, altri invece sotto forma di plugin, scritti in NASL (Nessus Attack Scripting Language). Nell'immagine sottostante viene mostrata l'interfaccia utente di Nessus, la quale è quella della versione gratis del prodotto.

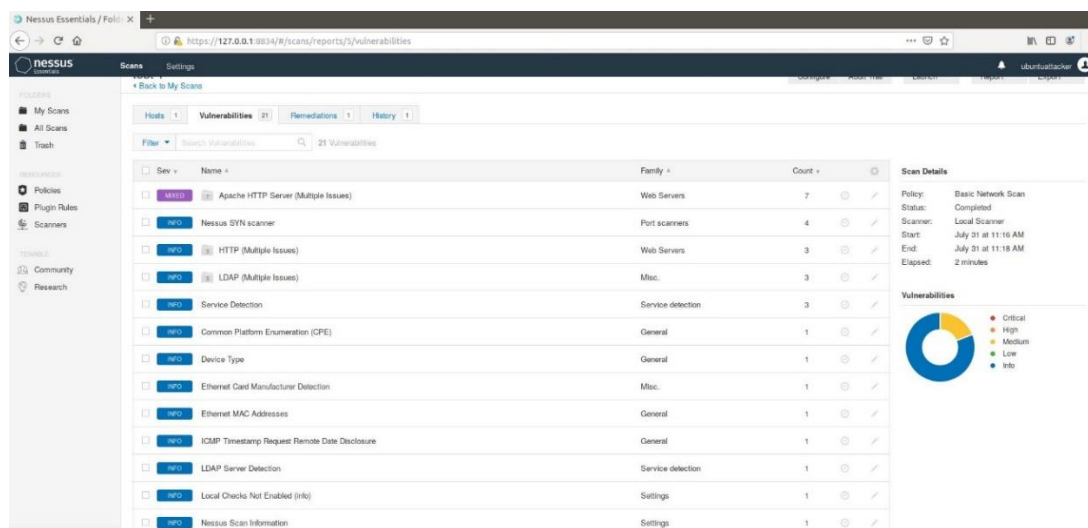


Essendo un software proprietario ha licenza a pagamento. Tuttavia, esiste una versione gratuita limitata alla possibilità di scannerizzare un massimo di 16 IP e con una ridotta quantità di test effettuabili rispetto alla versione completa.

Una volta avviata la scansione su Nessus, il software inizialmente effettua un port scan per ricercare porte aperte sulla macchina target avente IP indicato dall'utente. Una volta ottenuta la lista delle porte aperte il software tenta diversi exploit su di esse.

I risultati ottenuti vengono visualizzati sull'interfaccia grafica dell'utente ed è possibile scaricarli in diversi formati quali plain text, XML, HTML e LaTeX. Disabilitando l'opzione safe checks alcuni test di vulnerabilità cercheranno di attivare servizi vulnerabili o di mandare in crash il sistema operativo [22].

Successivamente l'installazione di Nessus, sono stati effettuati dei test automatici aventi come target il server su cui Jenkins è in esecuzione. Dai test sono emerse diverse problematiche, di cui la maggior parte considerate lievi da Nessus.



Come si evince dall'immagine contenente il grafico a torta che riassume le tipologie di vulnerabilità riscontrate, è stata riscontrata una serie di problematiche di media importanza legate ad Apache web server e una serie di informazioni che Nessus è riuscito ad ottenere sul sistema.

In particolare, il web server Apache, installato sul server, risulta non essere aggiornato all'ultima versione, nonostante in fase di installazione sia stato prelevato dal repository di Ubuntu. Anche dopo i test, provando ad aggiornare Apache dal repository, la versione attuale risultava l'ultima uscita. Le problematiche riscontrate su Apache sono state classificate con CVSS che partiva da 4.3 sino a 6.9 e comprendevano: Privilege escalation vulnerability, Access control bypass vulnerability e Denial of service.

Per risolvere le problematiche riscontrate con Apache è stata necessaria l'installazione dell'ultima versione disponibile del software prelevata non dal repository ma dall'esterno, scaricandola dalla rete.

Effettuando un ulteriore test successivamente all'aggiornamento di Apache non sono state più riscontrate problematiche di sicurezza.

Conclusa la fase di testing si passa ora alla threat analysis del sistema.

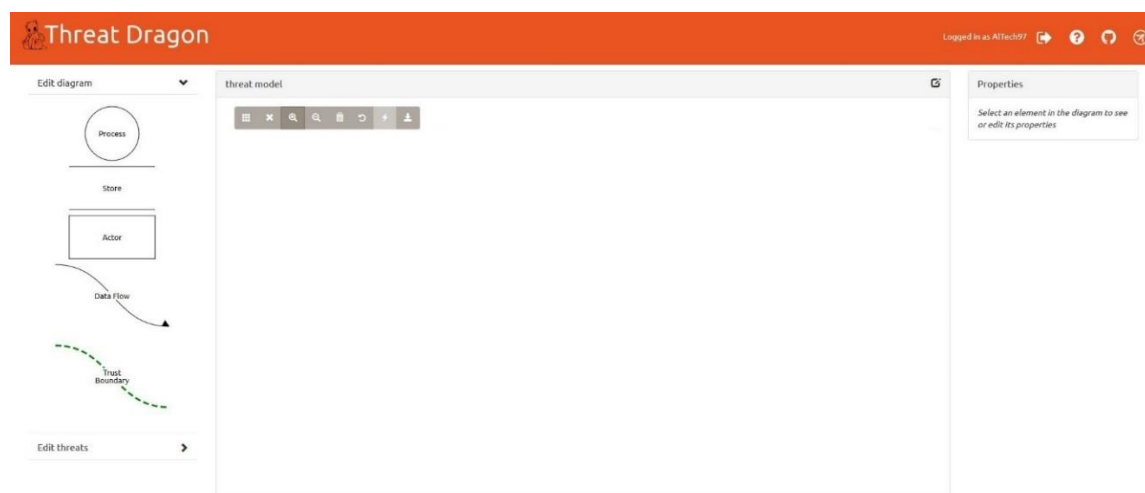
4.1.4. Threat analysis

Per la definizione dei rischi residui all'interno del sistema e per avere una sua “vista dall'alto” è stato sviluppato un suo threat model utilizzando OWASP Threat Dragon.

Threat dragon è un software open source prodotto da OWASP ed utilizzato per creare threat model di una qualsiasi applicazione.

Il software è disponibile sia in formato desktop, per Windows e OSX, sia via web, accedendo dal browser con il proprio account GitHub. La versione desktop è utilizzata principalmente quando si ha la necessità di creare un threat model di un sistema proprietario, poiché installandolo sulla propria macchina permette di mantenere la massima riservatezza delle informazioni inserite al suo interno, siccome il file generato viene salvato sul file system della propria macchina. La versione web è utilizzata principalmente per i sistemi open-source, in quanto permette il salvataggio e il caricamento dei diagrammi soltanto su repository pubbliche di GitHub.

L'interfaccia grafica è la stessa tra entrambe le versioni ed è la seguente:



A sinistra dell'interfaccia sono presenti due sezioni in colonna, la prima si chiama “Edit diagram” e viene utilizzata per l'aggiunta di nuovi elementi al diagramma, i quali sono:

Processo: rappresentato da un cerchio, indica un processo presente avviato su una macchina.

Store: ha nome incorniciato da due linee parallele sopra e sotto, rappresenta un qualsiasi deposito di dati, in locale o remoto.

Actor: ha come simbolo un rettangolo con all'interno il nome dell'attore. Un attore all'interno del sistema è una qualsiasi entità che interagisce con altre entità, ad esempio un browser web che interagisce con un server, o un server che interagisce con un altro server.

Data flow: rappresentato nel diagramma come una freccia, ha un nome e un verso e indica uno scambio di dati tra attore e attore o attore e processo. Un data flow può anche essere bidirezionale qualora vi sia uno scambio di dati tra le entità.

Trust boundary: indicato come una linea verde tratteggiata, rappresenta i confini del nostro sistema oppure una separazione tra diverse componenti dello stesso sistema che cooperano tra loro. Il trust boundary aiuta l'utente a comprendere la superficie d'attacco del sistema.

La seconda sezione presente a sinistra dell'interfaccia grafica, sotto "Edit diagram", chiamata "Edit threats", è utilizzata per l'aggiunta dei threat ad un elemento del diagramma.

I threat possono essere aggiunti manualmente o in automatico tramite l'intelligenza artificiale presente in Threat Dragon che consiglia l'aggiunta di alcuni threat.

Le tipologie di threat che possono essere aggiunte sono quelle del modello STRIDE.

New Threat

Title

A short title for the threat

STRIDE threat type

Threat status

Open Mitigated

Severity

High Medium Low

Description

Detailed description of the threat

Mitigations

Mitigations for the threat

Save Cancel

All'interno della scheda che compare quando si vuole aggiungere un nuovo threat sono presenti diversi campi da compilare:

Titolo: un breve titolo che rappresenta il threat. Il titolo verrà poi mostrato all'interno della lista.

Tipologia di threat: seguendo il modello STRIDE è presente come menu a tendina.

Stato del threat: selezionabile tra aperto, se non è ancora stato risolto o mitigated, se è stato risolto.

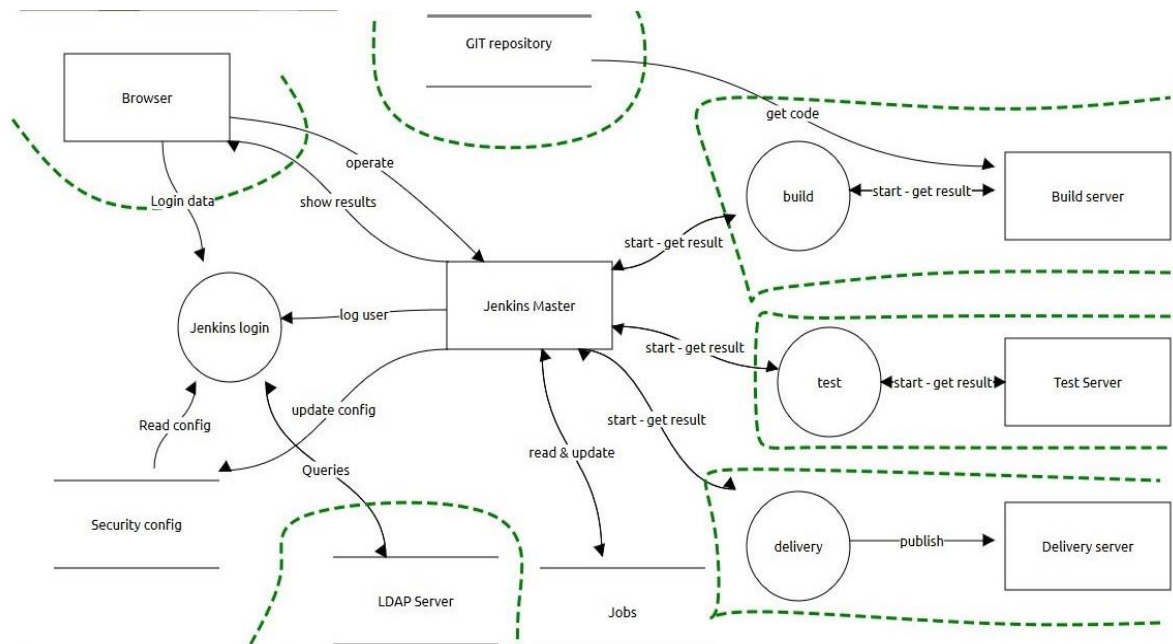
Gravità: classificata con livello alto, medio o basso.

Descrizione: in questo campo è possibile inserire una descrizione del threat.

Mitigazione: in questo campo di testo è possibile inserire una descrizione riguardo il come mitigare (se possibile) il threat indicato.

Al centro dell'interfaccia utente è presente l'area di lavoro, dove gli oggetti del diagramma possono essere spostati, collegati e rinominati. Sull'area di lavoro è presente inoltre una barra con diversi pulsanti utilizzati per effettuare diverse operazioni sul diagramma.

Il threat model del sistema è il seguente:



All'interno del threat model sono stati designati prima di tutto gli attori del sistema che sono:

Browser: Controllato dall'utente, permette l'accesso e l'interazione con il sistema dall'esterno.

Jenkins Master: rappresenta il nodo master di Jenkins, situato al centro del diagramma in quanto è lui che orchestra tutte le operazioni di CI/CD.

Build Server: rappresenta il nodo che ha il compito di effettuare le build del codice prelevato da Git.

Test Server: è il nodo che effettua i test sul codice precedentemente compilato. Il codice da testare viene passato al nodo master.

Delivery server: è l'ultimo nodo del sistema di sviluppo, il codice funzionante viene rilasciato su questo nodo e a partire da quest'ultimo distribuito e installato sul server del cliente.

Successivamente sono stati definiti gli storage del sistema e collegati ai relativi utilizzatori. Gli storage risultanti sono:

Configurazioni di sicurezza: è un file in cui sono salvate tutte le configurazioni di sicurezza di Jenkins, lette e modificate tramite il nodo master da utenti autorizzati.

Job list: è uno storage contenente la lista completa dei Job di Jenkins, viene letta e modificata dal nodo master.

LDAP Server: contiene i dati per l'accesso a Jenkins. Viene letto dal nodo master in fase di autenticazione dell'utente.

Git Repository: contiene tutto il codice in fase di sviluppo. Il codice viene scaricato dal Build server ogni qualvolta è disponibile un aggiornamento.

Infine, per completare il threat model sono stati inclusi i processi, legati ai diversi nodi del sistema. I processi inseriti sono:

Jenkins login: processo legato al nodo master. Permette di effettuare l'autenticazione dell'utente tramite la lettura delle configurazioni di sicurezza, il prelievo dei dati dal server LDAP e le credenziali di accesso fornite dall'utente tramite il browser.

Build process: è un processo situato sul server di build. Permette al server di interfacciarsi con il nodo master per avviare una build o modificare le configurazioni di quest'ultima. Una volta effettuata la build restituisce i risultati della build e il codice compilato al nodo master in modo che quest'ultimo possa mostrare dei report all'utente e inviare il codice al server di test.

Test process: servizio presente sul server di test, permette a quest'ultimo di interfacciarsi con il nodo master per avviare i test sul prodotto. Il codice da testare viene passato dal nodo master e anche i test da effettuare. Una volta conclusi i test il servizio restituisce un report, contenente i risultati, al master in modo che possa visualizzarli all'utente.

Delivery process: situato sul Delivery server, utilizzato anch'esso come interfaccia verso il nodo master, permette di caricare il codice compilato

e testato sul server di distribuzione, codice che poi sarà prelevato e distribuito al cliente.

Ultimato il threat model si passa alla fase di hardening del sistema per mitigare alcune problematiche di sicurezza.

4.2. Hardening di Jenkins

Su VirtualBox, installato in precedenza per gestire le diverse macchine virtuali previste, è stata creata la prima macchina, chiamata “Jenkins server”, configurata con Ubuntu Server 18.04.3 LTS. La macchina è stata oggetto dei test di sicurezza appena visti i quali hanno avuto come target anche Jenkins, che al momento dei test era presente sulla macchina alla versione 2.176.2 LTS.

In questo capitolo verranno descritte le attività effettuate per adoperare l’hardening di Jenkins. In particolare, verrà descritta la configurazione di LDAP, un server che permette la gestione dell’autenticazione separatamente da Jenkins. Successivamente verrà effettuata una separazione dei ruoli all’interno del sistema, inizialmente assente. Verrà aggiunto alla pipeline un software per il controllo automatico delle problematiche di sicurezza ed infine verranno mostrate ulteriori configurazioni di sicurezza del sistema.

4.2.1. Configurazione di LDAP

L’hardening di Jenkins ha avuto inizio con l’abilitazione della sicurezza al suo interno, nelle impostazioni generali, selezionando la casella “Enable security” in quanto di default è disattivata.

Successivamente è stata effettuata l’installazione e la configurazione di un server LDAP per la gestione degli accessi degli utenti a Jenkins.

LDAP è l’acronimo di Lightweight Directory Access Protocol. È un protocollo internet utilizzato da diversi programmi per ottenere

informazioni, generalmente dati di persone appartenenti ad una stessa organizzazione, da un server [23]. Il server LDAP permette di avere una struttura centralizzata e gerarchica da cui prelevare i dati da qualsiasi punto di accesso, se si dispone dei dovuti diritti definiti dall'amministratore del server. I dati vengono prelevati tramite query che, come per un database, permettono di richiedere anche un singolo dato appartenente ad una persona, oppure l'elenco di persone appartenente ad un gruppo.

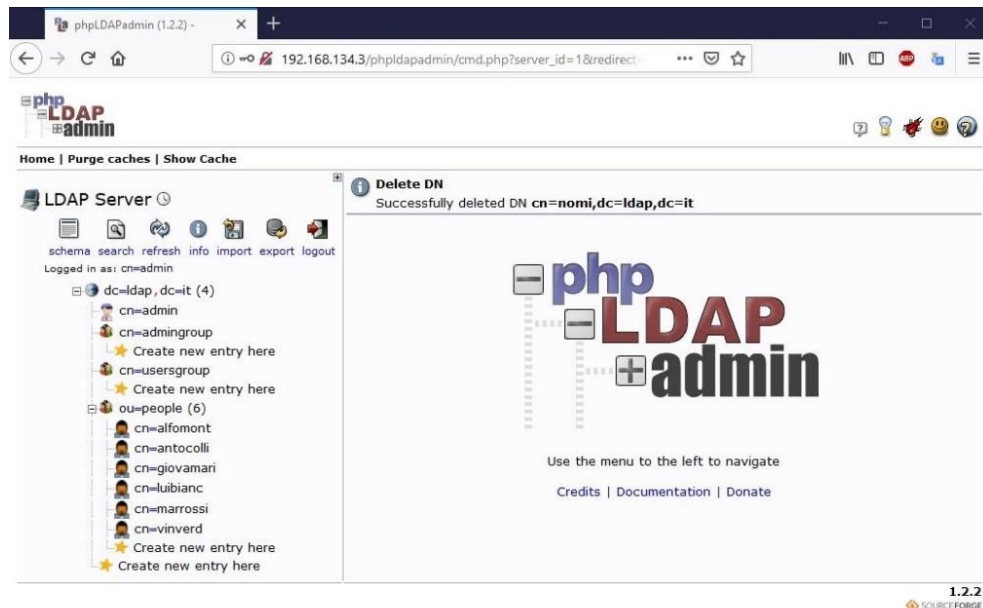
LDAP non è soltanto utilizzato per salvare dati riguardanti persone, è utilizzato anche per salvare certificati di crittografia, puntatori a stampanti ed altri servizi disponibili all'interno di una rete.

LDAP è un'alternativa snella a DAP, Directory Access Protocol, e richiede il protocollo di comunicazione TCP/IP. L'ultima versione del protocollo ha integrato il Simple Authentication and Security Layer, un framework utilizzato per l'autenticazione e la sicurezza dei protocolli Internet. LDAP si distingue da un classico database per il fatto che le operazioni compiute sul server LDAP sono ottimizzate in modo da occupare minor banda, dunque LDAP è la scelta migliore se si decide di implementare una struttura centralizzata su di un server [24].

Il server è stato configurato per memorizzare l'username, la password ed altre informazioni riguardanti gli utenti di Jenkins. LDAP garantisce che la memorizzazione delle informazioni e la gestione degli accessi venga effettuata in modo più sicuro rispetto a ciò che viene offerto da Jenkins di default.

In particolare, riguardo la configurazione del server LDAP, sono state definite due CN (common name) per diversificare le tipologie di utenti: *Admingroup*, contenente tutte le persone che saranno autorizzate a qualsiasi operazione all'interno di Jenkins, in quanto amministratori del sistema.

Usersgroup che comprende tutte le persone con diritti limitati in Jenkins, ad esempio sviluppatori che lavorano con Jenkins.



Successivamente è stato definito un OU (organizational unit) chiamato “people” a cui sono stati aggiunti diversi utenti, ognuno di essi appartenente ad usersgroup o admingroup. Gli utenti creati sono di tipo “Generic: user account”.

L'appartenenza da parte di un utente ad un determinato gruppo è determinata dal valore dell'attributo “gidNumber”, un'intero. Il valore dell'attributo all'interno del campo gidNumber nella scheda dell'utente deve essere pari al valore dell'attributo gidNumber presente nel CN di riferimento.

La creazione di utenti e gruppi è stata effettuata tramite l'interfaccia grafica del server LDAP disponibile alla pagina phpldapadmin, dopo aver effettuato l'accesso con le credenziali dell'amministratore, definite in fase di installazione e configurazione iniziale.

Una volta creati i due gruppi sono stati creati dei nuovi utenti cliccando sul link “create new entry here” presente sotto il nome del gruppo.

Dopo aver configurato il server LDAP è stato configurato Jenkins affinché utilizzasse LDAP come “database delle password”.

In particolare, è stato inserito in Jenkins “LDAP plugin” e successivamente nella sezione “Security Realm”, presente nella pagina di configurazione

della sicurezza globale, selezionando il radio button “LDAP”. Sono stati compilati i campi della form riguardanti l’indirizzo del server LDAP, root DN, che rappresenta la radice dell’albero di LDAP, il manager DN, ovvero l’amministratore del server LDAP e la sua password.

Prima di salvare la configurazione è Jenkins permette di testare il funzionamento del server mediante l’apposito tasto “Test LDAP settings” il quale farà comparire una form di login per evitare all’amministratore del sistema di restare “chiuso fuori” una volta applicate le modifiche.

Nell’immagine sottostante è possibile vedere la configurazione effettuata su Jenkins e appena spiegata.

Configura la sicurezza globale

☒ Enable security
☐ Disable remember me
Access Control

Security Realm

☐ Database degli utenti interno di Jenkins
☐ Delega al container servlet
☒ LDAP

Server

Server: ldap://192.168.134.3

root DN: dc=ldap,dc=it

☐ Allow blank rootDN

User search base:

User search filter: uid=[0]

Group search base:

Group search filter:

Group membership: ☐ Parse user attribute for list of LDAP groups
☒ Search for LDAP groups containing user
Group membership filter:

Manager DN: cn=admin,dc=ldap,dc=it

Manager Password:

Display Name LDAP attribute: displayname

Email Address LDAP attribute: mail

Environment Properties:

Ignore if Unavailable: ☐

Authorization

☐ Unix user/group database
☐ Chiunque può fare qualsiasi cosa
☐ Gli utenti che hanno eseguito l'accesso possono compiere qualunque azione
☐ Matrix-based security
☐ Modalità legacy

4.2.2. Definizione dei ruoli nel sistema

Per la definizione dei ruoli all'interno del sistema è stato utilizzato il plugin "Project-based matrix authorization strategy".

Questo plugin di Jenkins permette di definire, attraverso due diverse matrici, dei permessi a grana fine sull'accesso ai job di Jenkins da parte di singoli utenti e gruppi presenti all'interno del server LDAP. I permessi possono essere definiti sia nella sezione “configura la sicurezza globale”, dove viene definita una matrice delle operazioni concesse ad ogni utente o gruppo specificato dall'utente:

Modalità legacy

☒ Project-based Matrix Authorization Strategy

User/group	Generali	Credentials	Agente	Processo	Esegui	Visualizza	Sistema di controllo del codice sorgente	Lockable Resources																							
	Administer	Read	Create	Delete	Update	View	Build	Cancel	Configure	Create	Delete	Disconnect	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Delete	Workspace	Replay	Update	Configure	Create	Delete	Read	Tag	Reserve	Unlock
Anonymous Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Authenticated Users	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
admingroup	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
amontesi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
usersgroup	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
vverdi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Add user or group...

Role-Based Strategy

Markup Formatter

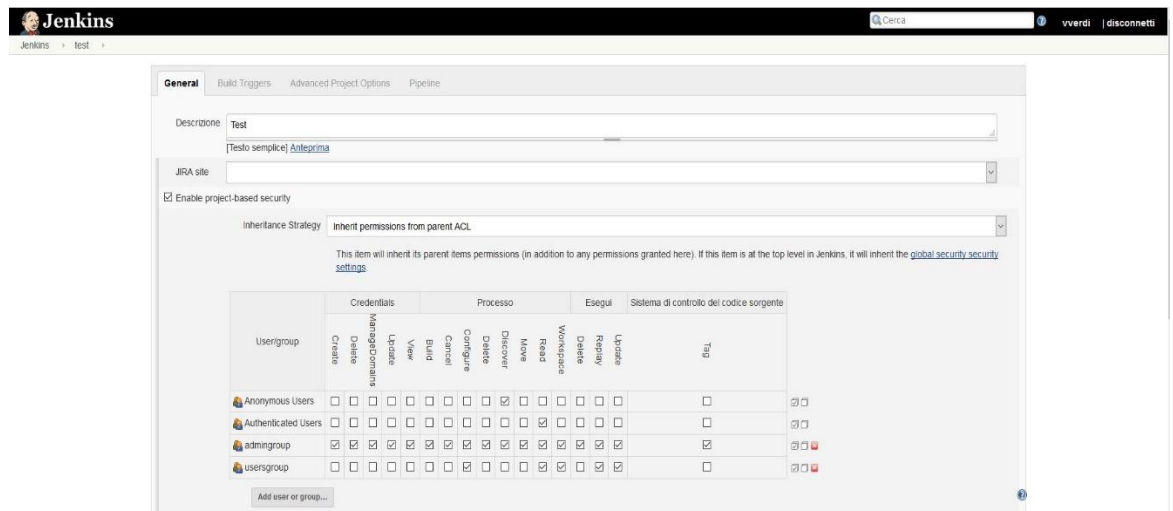
Markup Formatter

Testo semplice

Considera l'intero input come testo semplice. I caratteri non sicuri HTML come < ed & sono sottoposti ad escape e sostituiti dalle rispettive entità carattere.

Nella matrice sono presenti di default due gruppi “Anonymous Users” e “Authenticated Users”, non eliminabili, ai quali è possibile attribuire o togliere dei permessi. Ai gruppi di default è possibile aggiungerne altri o aggiungere singoli utenti. I gruppi o i singoli utenti aggiunti devono essere realmente esistenti all'interno di LDAP altrimenti il plugin non ne permetterà l'aggiunta.

Al momento della creazione di un nuovo progetto in Jenkins, tramite questo plugin, è possibile definire una seconda matrice per raffinare la grana dell'assegnazioni di permessi a gruppi o singoli utenti.



Questa ulteriore matrice è attivabile inserendo la spunta sulla casella “Enable project-based security” nella sezione “general”, sulla pagina che viene mostrata dopo aver creato un nuovo progetto.

Anche in questa matrice è possibile aggiungere utenti o gruppi a quelli inseriti di default, l’inserimento sarà possibile solo se i gruppi e gli utenti aggiunti risulteranno esistenti all’interno del server LDAP.

4.2.3. Controllo del codice

Per inserire la funzionalità di controllo automatico della presenza di bug, cattiva programmazione e falle di sicurezza all’interno del codice è stato prelevato da Git Hub un progetto java open source.

È stata creata una nuova pipeline al cui interno è stato caricato il codice proveniente dal repository.

Di default la pipeline presenta tre stadi di esecuzione: Preparation, Build e Results. A questi tre stadi ne è stato aggiunto un’ulteriore, dedicato a SonarQube, inserito tra quello di Preparation e quello di Build. Lo script della pipeline ora risulta essere il seguente:

```
node {
  def mvnHome
    // for display purposes
```

```

stage('Preparation') {
    // Get some code from a GitHub repository
    git 'https://github.com/ZUGFeRD/mustangproject.git'
    // Get the Maven tool.
    mvnHome = tool 'M3'
}

stage('Sonarqube'){
    // Run the sonar static analisys
    withEnv(["MVN_HOME=$mvnHome"]) {
        if (isUnix()) {
            sh "$MVN_HOME/bin/mvn"    sonar:sonar    -Dsonar.pro-
            jectKey=mustang-Dso-
            nar.host.url=http://192.168.134.3:9000    -Dsonar.lo-
            gin=6c4867dcaa1190507c9007f07e8a3ed934d4da6c'
        } else {
            //Do nothing
        }
    }
}

stage('Build') {
    // Run the maven build
    withEnv(["MVN_HOME=$mvnHome"]) {
        if (isUnix()) {
            sh "$MVN_HOME/bin/mvn" -Dmaven.test.failure.ignore
            clean package'
        } else {
            bat("/"%MVN_HOME%\bin\mvn"    -Dmaven.test.fai-
            lure.ignore clean package/)
        }
    }
}

stage('Results') {

```

```

        junit '**/target/surefire-reports/TEST-*.xml'
        archiveArtifacts 'target/*.jar'
    }
}

```

Il codice appena visto viene inserito nella sezione “pipeline script” nelle impostazioni generali della pipeline creata. Il pipeline script viene scritto nel linguaggio Groovy.

Verrà ora introdotto SonarQube.

SonarQube è un analizzatore di codice che usa tecniche avanzate per la ricerca di bug, di cattive pratiche di programmazione e vulnerabilità del software [25]. È disponibile per diverse piattaforme e IDE tra cui Eclipse, per gli sviluppatori. SonarQube permette di analizzare codice scritto in diversi linguaggi e per ogni linguaggio ha molte regole che riguardano sicurezza e buona programmazione [26]. L’analisi viene effettuata leggendo tutto il codice dato in input e verificando se rispetta tutte le regole. Se inserito all’interno di una pipeline di Jenkins, all’interno di un proprio stage o all’interno di uno stage già esistente, permette di effettuare l’analisi in modo del tutto automatico.

L’architettura della piattaforma di SonarQube è formata da quattro componenti [27]:

1. SonarQube server:

È unico e avvia tre processi:

Web server per sviluppatori e manager per ricercare snapshot e configurare l’istanza di SonarQube.

Server di ricerca, basato su Elasticsearch ed utilizzato per la ricerca nell’interfaccia utente.

Server del motore di computazione, si occupa dell’analisi del codice e dello storage dei risultati nel database.

2. SonarQube database

Utilizzato per contenere informazioni sulla configurazione dell’istanza di SonarQube e gli snapshot dei progetti, viste ecc.

3. Plugins

Sono installati sul server e vengono utilizzati per ampliare il programma aggiungendo nuove funzionalità.

4. Uno o più SonarScanner

Vengono eseguiti sul server di Continuous Integration e sono utilizzati per analizzare il progetto in input.

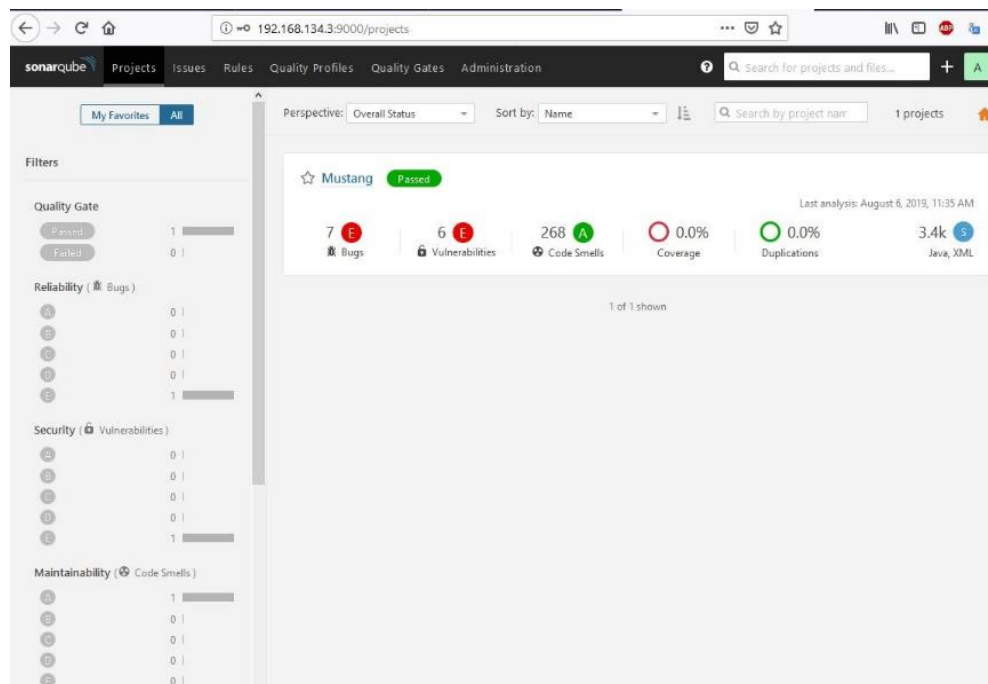
L'integrazione di SonarQube all'interno del sistema avviene mediante il seguente flusso di eventi:

Gli sviluppatori scrivono il codice all'interno di un'IDE. Successivamente viene effettuato il push del codice in GIT.

L'evento del push viene intercettato dal Continuous Integration server che effettua la build automatica del codice aggiornato e esegue il SonarScanner, necessario per effettuare l'analisi con SonarQube.

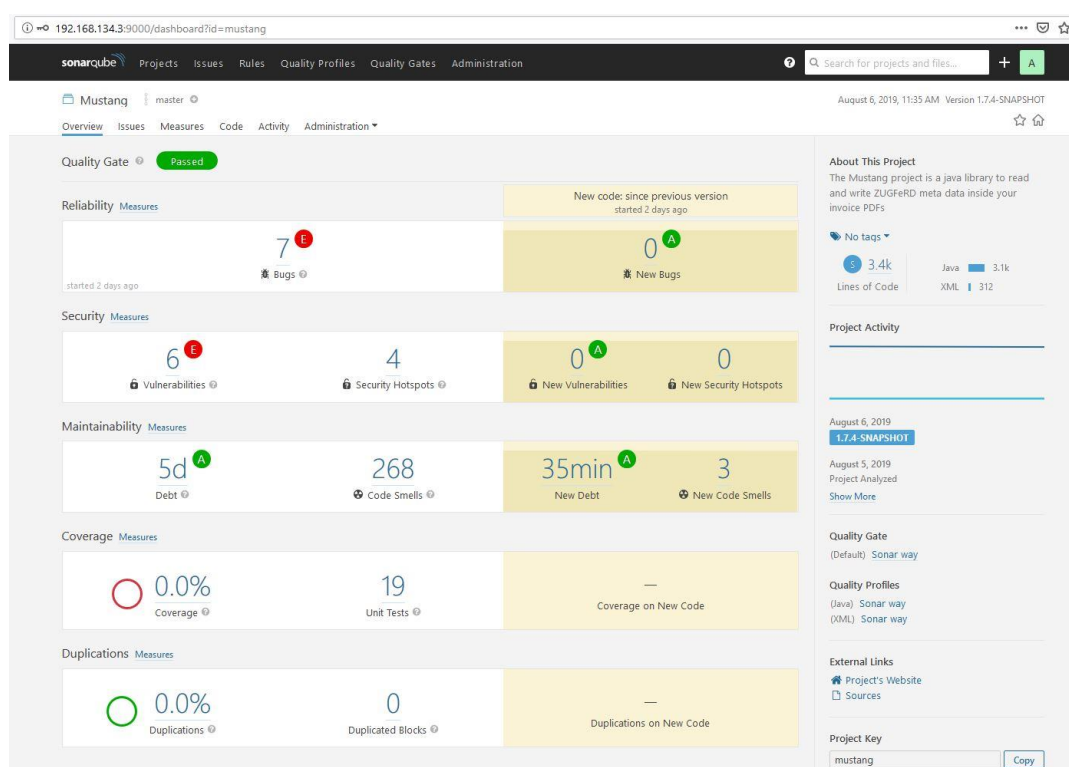
Il risultato dell'analisi viene inviato al server di SonarQube e viene processato.

Il risultato viene salvato nel database e mostrato nell'interfaccia grafica dell'utente.



Gli sviluppatori visualizzano il report e cercano di risolvere le problematiche indicate in maniera dettagliata, avendo a disposizione anche suggerimenti per gli errori noti, e la visualizzazione delle righe di codice che generano quella problematica.

Una volta avviata la pipeline, accedendo a SonarQube tramite la porta a lui dedicata, ci viene mostrato un report contenente bug, vulnerability e code smells.



Il report risulta essere molto dettagliato poiché posizionandosi sulla pagina “Issues” è possibile visionare un elenco completo di tutte le problematiche riscontrate. Per ogni problematica è indicata la riga dove quest’ultima è stata riscontrata e una possibile risoluzione. Le vulnerabilità del software vengono inoltre suddivise in base alla loro importanza.

The screenshot shows the SonarQube web interface for the 'Mustang' project. The left sidebar contains filters for 'Type' and 'Severity'. The 'Type' filter shows 'Vulnerability' with 6 items. The 'Severity' filter shows 'Critical' with 1 item and 'Minor' with 4 items. The main panel displays a list of vulnerabilities. The first two are 'Make namespace a static final constant or non-public and provide accessors if needed.' and 'Make prefix a static final constant or non-public and provide accessors if needed.', both with a severity of 'Minor' and an effort of '10min'. The next two are 'Use a logger to log this exception.' with a severity of 'Minor' and an effort of '10min'. The last one is 'Secure this "Transformer" by either disabling external DTDs or enabling secure processing.' with a severity of 'Critical' and an effort of '5min'. The bottom of the list shows 'Disable XML external entity (XXE) processing.' with a severity of 'Blocker' and an effort of '15min'.

4.2.4. Ulteriori configurazioni di sicurezza

Per ultimare l'hardening di Jenkins sono state effettuate ulteriori configurazioni di sicurezza. In particolare, è stata disabilitata la CLI (command line interface) per ridurre la superficie d'attacco. Nel caso d'uso del progetto la CLI risulta un servizio inutile in quanto per effettuare operazioni su Jenkins basta autenticarsi tramite la sua interfaccia grafica. L'autenticazione infatti, gestita tramite LDAP, permette la ripartizione dei ruoli del sistema e dei relativi permessi a compiere determinate azioni. È stato disabilitato l'SSHD deamon, siccome è stato deciso di operare sull'interfaccia grafica e non tramite CLI. L'SSHD deamon è un servizio che permette la comunicazione con Jenkins tramite protocollo SSH, disattivandolo è stata ridotta ulteriormente la superficie d'attacco.

In generale, disabilitare i servizi inutilizzati e connessi alla rete aiuta a ridurre la possibilità di ricevere attacchi di qualsiasi tipo.

4.3. Valutazione dei rischi residui.

Dopo aver effettuato i test di sicurezza tramite OWASP ZAP e Nessus è stato rappresentato il threat model del sistema che ha permesso una vista ad ampio spettro di tutto l'ambiente che è stato realizzato. Successivamente è stato effettuato l'hardening del sistema migliorando molti aspetti di sicurezza e tralasciandone alcuni.

I rischi residui all'interno del sistema sono quelli segnalati da OWASP ZAP in quanto non rappresentano un rischio grave per l'intero sistema ed inoltre il rilascio frequente di aggiornamenti di Jenkins permette la risoluzione delle problematiche riscontrate in breve tempo.

In particolare, le problematiche residue nel sistema sono:

- L'opzione X-Frame non presente nell'header delle pagine di Jenkins.
- Intestazione X-Content-Type mancante.
- Intestazione X-XSS-Protection mancante.
- Rivelazione di informazioni tramite il messaggio d'errore generato.
- Assenza del token anti-CSRF
- Assenza del flag HttpOnly all'interno dei cookie scambiati.

Capitolo 5: Conclusioni e sviluppi futuri

Dopo aver affrontato le diverse problematiche di cui la metodologia DevOps soffre, e applicato DevSecOps, è possibile affermare che quest'ultima risulta essere la scelta migliore per le grandi aziende che intendono sviluppare e rilasciare parti del prodotto in modo sicuro veloce. L'applicazione di questa metodologia inoltre non impatta sul time to market del prodotto in quanto i test inseriti all'interno della pipeline di sviluppo sono di tipo automatico richiedono poco tempo per la loro esecuzione. Inoltre, questi test permettono, tramite dei report molto dettagliati mostrati all'interno di questa tesi, di arginare in modo rapido le principali problematiche di sicurezza.

La fase di test di sicurezza effettuati sul sistema che adottava il metodo DevOps ha messo in risalto tutte le vulnerabilità di questa metodologia, le quali sono state risolte in parte, migliorando molti aspetti dell'intero ambiente di sviluppo.

Gli strumenti utilizzati nelle fasi di test di sicurezza e di hardening del sistema basato su DevOps sono inoltre risultati molto validi e applicabili in grandi realtà come Ericsson in quanto permettono di ottenere dei report molto dettagliati.

Lo scopo di questa tesi è stato quello di mostrare al lettore le modalità di applicazione di DevSecOps, metodologia che permette di avere un ambiente di sviluppo e rilascio che bada alla sicurezza su diversi aspetti e che quest'ultima non rappresenta affatto un rallentamento per l'intero sistema, come si poteva pensare in principio.

Come accade con molte sperimentazioni, anche questa descritta nella tesi, che comprende l'implementazione della metodologia DevSecOps all'interno di un sistema DevOps, ha avuto alcuni problemi durante la sua esecuzione.

Inizialmente, nella fase di test, era stato previsto anche l'utilizzo di OpenVAS, un software simile a Nessus ed OWASP ZAP che permette di analizzare le criticità di sicurezza relative ad una macchina connessa alla rete, fornendo soltanto l'indirizzo IP. Il problema riscontrato dopo l'installazione e la

configurazione di OpenVAS è dovuto alla restituzione di report vuoti da parte del software, il quale indicava che la possibile causa di questa risposta fosse l'irraggiungibilità dell'indirizzo IP dato, quando in realtà l'indirizzo risultava raggiungibile tranquillamente dalla macchina su cui OpenVAS veniva lanciato.

Un altro problema riscontrato è stato il mancato ritrovamento del file contenente gli header di Jenkins, diversi tentativi sono stati effettuati senza successo, dunque non è stato possibile modificare il valore di alcuni header delle pagine di Jenkins. In ogni caso questa problematica ha avuto poco impatto all'interno della sperimentazione siccome la modifica degli header non rientrava tra le priorità.

I possibili sviluppi futuri di questo progetto riguardano il miglioramento del livello di sicurezza apportato al sistema. In particolare, può risultare utile inserire un algoritmo di cifratura che cripti le informazioni prima che queste vengano trasferite da un nodo all'altro di Jenkins, in modo che chiunque riesca ad ottenere tali informazioni non sarà in grado di decifrarle.

Un ulteriore sviluppo futuro riguarda l'inserimento di un controllo che verifichi l'attendibilità del software di terze parti eventualmente utilizzato in combinazione con software sviluppato in DevOps. Software che essendo utilizzato con la metodologia DevOps necessiterà di essere aggiornato e controllato in maniera automatica.

Glossario

A

Asset: Dal punto di vista informatico si intendono le risorse informatiche aziendali (server, workstation ecc.).

Attacco: Tentativo di violazione della sicurezza tramite lo sfruttamento (exploitation) di una vulnerabilità.

B

Bug: Errore di implementazione dell'asset.

C

Continuous Delivery (CD): estende l'integrazione continua distribuendo le modifiche al codice alla fase di testing e/o alla fase di produzione. Consente di automatizzare il testing anche grazie al cloud che permette di creare e/o replicare, ad un costo minore, un ambiente di testing

Continuous Integration (CI): Per integrazione continua si intende la creazione di build e test eseguiti in automatico per ridurre il tempo richiesto per la convalida e la pubblicazione di aggiornamenti software.

E

Exploit, exploitation: procedura atta a sfruttare una vulnerabilità, causa un comportamento inatteso in un asset, trasforma una minaccia in realtà.

F

Falla di sicurezza: problematica di sicurezza che permette ad un hacker di attaccare il sistema.

Fault: condizione anormale o un difetto di un elemento del sistema che può comportare ad un guasto o ad un faliure.

Faliure: Comportamento del software diverso dai requisiti di progetto.

H

Hacker: è un esperto nell'ambito informatico, soprattutto di sicurezza informatica, non necessariamente malevolo. Un hacker viene classificato in diverse categorie per distinguerne i tipi:

Cracker: programmatori specializzati nell'infrangere sistemi di sicurezza per rubare o distruggere dati

Script kiddie: cracker che adoperano script scritti da altri, non sapendoli produrre. In genere sono hacker alle primissime esperienze

Hackivist: Sfruttano gli attacchi DDoS (Distributed Denial of Service) per mandare offline il sito di un'azienda con etica cattiva.

Phracher: Hacker che opera nell'ambito della telefonia, può rubare programmi che offrono servizi telefonici o attaccare computer e database di società telefoniche.

Black hat: hacker con cattive intenzioni che sfrutta le proprie abilità per delinquere.

White hat: hacker buono, sfrutta le proprie abilità per aumentare le difese di un sistema.

Grey hat: è una via di mezzo tra black hat e white hat. Effettua operazioni a danno di qualcuno o a favore di altri a scopo economico.

Hardening: Attività che consiste nel configurare una macchina in modo che sia difficile da espugnare. Agisce principalmente sulla configurazione del sistema, servizi, applicazioni, utenze, privilegi ecc

I

Intrusione: un attacco che ha avuto successo.

M

Meccanismo di sicurezza: ciò che ha lo scopo di far rispettare le policy. Sono divisi in tre diverse categorie:

- Meccanismi di prevenzione: Impediscono le interazioni tra un asset e un utente, troppa prevenzione non permette l'interazione nemmeno con gli utenti normali del sistema, per cui è necessario che l'asset venga esposto alla rete aprendo alcune porte TCP
- Meccanismi di rivelazione: controllano le interazioni tra un asset e un utente, consistono in un controllo del traffico sulle porte TCP aperte e controllo degli input passati a una funzione.
- Meccanismi di reazione: utilizzati per ripristinare il sistema a seguito di un incidente.

Microservizio: Architettura software che permette di realizzare un'applicazione basata su un gruppo di servizi indipendenti, di piccole dimensioni, connessi tra loro e che comunicano mediante interfaccia (API basata su http).

Minaccia (Threat): insieme di circostanze potenzialmente pericolose, in termini di violazione del sistema (o possibile tale). Si divide in diverse classi:

- rivelazione (accesso non autorizzato a dati)
- inganno (il sistema accetta un dato falso),
- interruzione (DoS), usurpazione (controllo non autorizzato del sistema (o parte))
- Privilege escalation, chiarito successivamente.
- Ripudio (negazione dell'esecuzione di azioni)

P

Privilege escalation: guadagnare l'accesso a risorse normalmente riservate ad utenti con privilegi (o diritti di accesso). È un attacco andato a buon fine.

Policy: Insieme di regole che stabiliscono quali soggetti hanno quali diritti su quali oggetti.

R

Root compromise: Situazione in cui l'hacker ha ottenuto il controllo totale della macchina attaccata.

S

Safety: relativa a incidenti causati da eventi accidentali

Security: relativa ad incidenti causati volontariamente

U

User-experience: per user experience si intende ciò che una persona prova durante l'utilizzo di un prodotto. Include aspetti percettivi quali l'utilità, la semplicità di utilizzo e l'efficienza del sistema.

V

Vulnerabilità: Problema HW, SW, di configurazione o di procedura che rende possibile l'uso improprio di dati o risorse HW e SW.

Bibliografia

- [1] A. Aijaz, «Release Frequency: A Need for Speed,» DZone, [Online]. Available: <https://dzone.com/articles/release-frequency-a-need-for-speed>.
- [2] Ericsson, «5G security - enabling a trustworthy 5G system,» [Online]. Available: https://www.ericsson.com/en/white-papers/5g-security---enabling-a-trustworthy-5g-system?gclid=EAIaIQobChMir7_IruzI5AIVGKqaCh1Gmg2gEAAYASAAEgL1uvD_BwE.
- [3] T. Smith, «DevSecOps Keys to Success,» DZone, [Online]. Available: <https://dzone.com/articles/devsecops-keys-to-success>.
- [4] Amazon, «what is devops,» [Online]. Available: <https://aws.amazon.com/it/devops/what-is-devops/>.
- [5] A.Salgarelli, «DevOps, definizione e differenze con agile,» [Online]. Available: <https://medium.com/4devops/metodo-agile-e-metodologia-devops-definizioni-e-differenze-2b4aeb87ee0f>.
- [6] Wikipedia, «Metodologia agile,» [Online]. Available: https://it.wikipedia.org/wiki/Metodologia_agile.
- [7] Agile, «manifesto,» [Online]. Available: <https://agilemanifesto.org/iso/it/principles.html>.
- [8] wikipedia, «Jenkins,» [Online]. Available: [https://it.wikipedia.org/wiki/Jenkins_\(software\)](https://it.wikipedia.org/wiki/Jenkins_(software)).
- [9] scmGalaxy, «Jenkins architecture explained,» [Online]. Available: <http://www.scmgalaxy.com/tutorials/jenkins-architecture-explained/>.

- [10] wikipedia, «Git,» [Online]. Available:
[https://it.wikipedia.org/wiki/Git_\(software\)](https://it.wikipedia.org/wiki/Git_(software)).
- [11] wikipedia, «GitHub,» [Online]. Available:
<https://it.wikipedia.org/wiki/GitHub>.
- [12] techcrunch, «What Exactly Is GitHub Anyway?,» [Online].
Available: <https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/?guccounter=1>.
- [13] N. Boldrini, «DevOps: migliorare la sicurezza in 3 mosse,» [Online].
Available: <https://www.zerounoweb.it/software/sviluppo-software/devops-migliorare-la-sicurezza-in-3-mosse/>.
- [14] M.Bishop, «1.2 Threats,» in *Introduction to computer security*, Addison-Wesley Professional.
- [15] N. Shevchenko, «Threat Modeling: 12 Available Methods,» [Online].
Available: https://insights.sei.cmu.edu/sei_blog/2018/12/threat-modeling-12-available-methods.html.
- [16] M.Bishop, «1.1 The basic components,» in *Introduction to computer security*, Addison-Wesley Professional.
- [17] A.Piva, «La sicurezza informatica tra confidenzialità, disponibilità e integrità dei dati,» [Online]. Available:
https://blog.osservatori.net/it_it/sicurezza-informatica-disponibilit%C3%A0-e-integrit%C3%A0-dei-dati.
- [18] Red Hat, «Cos'è la metodologia DevSecOps?,» [Online]. Available:
<https://www.redhat.com/it/topics/devops/what-is-devsecops>.
- [19] devsecops, «Manifesto,» [Online]. Available:
<https://www.devsecops.org/>.

- [20] R.Gallazzi, «OWASP ZAP: un potente strumento per scoprire vulnerabilità di siti Web,» [Online]. Available: <https://www.guruadvisor.net/it/sicurezza/825-owasp-zap-un-potente-strumento-per-scoprire-vulnerabilita-di-siti-web>.
- [21] wikipedia, «Nessus,» [Online]. Available: <https://it.wikipedia.org/wiki/Nessus>.
- [22] L.Obbayi, «A Brief Introduction to the Nessus Vulnerability Scanner,» [Online]. Available: <https://resources.infosecinstitute.com/a-brief-introduction-to-the-nessus-vulnerability-scanner/#gref>.
- [23] wikipedia, «Lightweight Directory Access Protocol,» [Online]. Available: https://it.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol.
- [24] gracion, «What is LDAP?,» [Online]. Available: <https://www.gracion.com/server/whatldap.html>.
- [25] wikipedia, «SonarQube,» [Online]. Available: <https://en.wikipedia.org/wiki/SonarQube>.
- [26] M.Nadeem, «Why SonarQube: An Introduction to Static Code Analysis,» [Online]. Available: <https://dzone.com/articles/why-sonarqube-1>.
- [27] SonarQube, «Architecture and Integration,» [Online]. Available: <https://docs.sonarqube.org/latest/architecture/architecture-integration/>.

Ringraziamenti

Un grande ringraziamento va alla mia famiglia, in particolare ai miei genitori per avermi sempre sostenuto ed avermi dato la possibilità di proseguire con gli studi e raggiungere questo primo traguardo importante.

Ringrazio il mio relatore, il professore Salvatore La Torre, per la disponibilità.

Ringrazio la sede Ericsson di Pagani per avermi concesso la stupenda opportunità di crescita personale e professionale tramite il tirocinio curriculare. Ci tenevo a ringraziare in particolar modo i miei relatori, Claudio e Giuseppe, e tutto il team di DevOps, con cui ho effettuato il tirocinio, formato da: Claudio, Paolo, Elio, Antonio, Giovanni, Arcangelo ed Anna. Ringrazio anche Francesco e Francesca per la loro disponibilità.

Ringrazio Annalisa, la mia ragazza, la quale mi è sempre stata d'aiuto moralmente, facendomi credere in me stesso e aiutandomi nei momenti di difficoltà.

Ringrazio inoltre i miei colleghi universitari: Alessandro, Cosimo, Luca, Alfonso, Bernardino, Christian, Ivan, Giammaria e Valeria per tutto il supporto che mi hanno dato durante questi anni di studio.