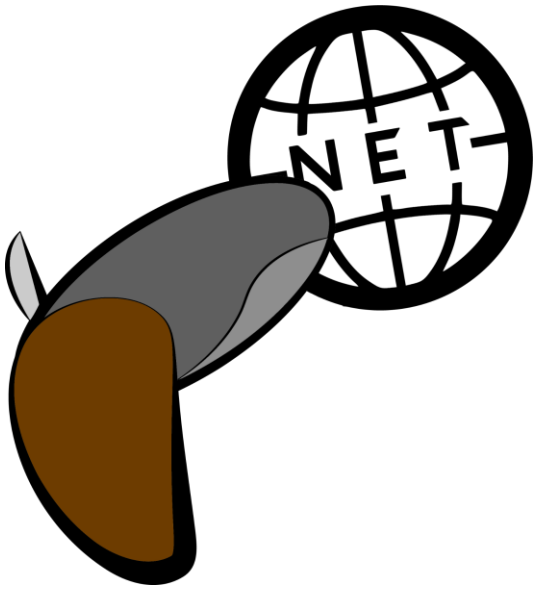


SDD
System Design Document

NetGun

Versione	2.0
Data	02/02/2023
Destinatario	Professore Carmine Gravino
Presentato da	Carlo Colizzi, Giulio Incoronato, Antonio Mazzarella

Team Members

Nome	Informazioni di contatto
Carlo Colizzi	c.colizzi@studenti.unisa.it
Giulio Incoronato	g.incoronato2@studenti.unisa.it
Antonio Mazzarella	a.mazzarella5@studenti.unisa.it

Revision History

Data	Versione	Descrizione	Autori
1/12/2022	0.1	Stesura della sezione revision history, team members e del Sommario	**Giulio Incoronato**
2/12/2022	0.2	Stesura dell'introduzione e dello scopo del sistema	**Carlo Colizzi**
3/12/2022	0.3	Definizione degli obiettivi di design e completamento del primo paragrafo	**Tutto il gruppo**
4/12/2022	0.4	Definizione dell'architettura del sistema	**Tutto il gruppo**
5/12/2022	0.5	Stesura del secondo paragrafo e inizio decomposizione del sistema	**Carlo Colizzi, Giulio Incoronato**
6/12/2022	0.6	Decomposizione e diagramma delle componenti completate	**Carlo Colizzi, Giulio Incoronato**
7/12/2022	0.7	Stesura e definizione del	**Carlo Colizzi**

		mapping software/hardware	
8/12/2022	0.8	Aggiunta del Deployment Diagram	**Tutto il gruppo**
8/12/2022	0.9	Aggiunta dell'albero descrittore dei file persistenti	**Giulio Incoronato**
9/12/2022	1.0	Definizione del controllo globale del software e delle condizioni limite	**Carlo Colizzi, Giulio Incoronato**
9/12/2022	1.1	Verifica e correzione degli errori	**Tutto il gruppo**
10/12/2022	1.1	Aggiunto un diagramma sui servizi dei sottosistemi	**Antonio Mazzarella, Giulio Incoronato**
11/12/2022	1.2	Chiusura del paragrafo 4 e aggiunta del glossario	**Giulio Incoronato**
15/11/2022	1.3	Verifica e correzione degli errori	**Tutto il gruppo**
02/02/2023	2.0	Revisione del documento	**Giulio Incoronato, Carlo Colizzi**

Sommario

1 Introduzione

1.1 Scopo del Sistema

NetGun ha l'obiettivo di essere un Framework per il Penetration Testing (Testing Black Box di infrastrutture in rete).

È possibile racchiudere il sistema il 3 componenti principali. La componente per lo scanning, la componente per l'enumerazione dei dati raccolti, e le utilities che assistono l'utente in tutte le fasi del pre e post scanning.

Inoltre, ha il fine di facilitare una pratica complessa come i Penetration Test, così da permettere ai PT di concentrarsi su aspetti più delicati, automatizzando e velocizzando i task alla base di questo tipo di Testing.

1.2 Design Goals

In questa sezione si andranno a presentare i Design Goals, ovvero le qualità sulle quali il sistema deve essere focalizzato, qualsiasi decisione di Design fatta, avrà come cardine il rispetto dei Design Goal sotto descritti.

I Design Goal del Progetto sono divisi nelle seguenti categorie:

- Performance
- Modificability
- Legibility
- Robustness
- Portability

Rank	ID Design Goal	Descrizione	RNF di origine
Categoria: Performance			
3	DG_1 Velocità di accesso ai dati persistenti	Il sistema dovrà permettere un tempo di accesso ai dati persistenti minore di un secondo	RNF_9
8	DG_2 Tempo di Risposta dello Scanner non deterministico	Il sistema non può garantire un tempo di risposta deterministico per ogni Scan, a causa dell affidabilità della rete	RNF_2
1	DG_3 Incremento del Throughput	Il sistema deve permettere un incremento del Throughput, aumentando il numero di richieste inviate al server in un intervallo di tempo (Tutto ciò in un ambiente virtualizzato).	RNF_1
Categoria: Modificability			
4	DG_4	Il sistema deve	RNF_3

	Manutenzione e Miglioramento facilitati	permettere una facile modificabilità data dalla modularità delle sue componenti. Si intende realizzare questo goal tramite un forte disaccoppiamento dei sottosistemi.	
Categoria: Leggibility			
5	DG_5 Leggibilità del codice	Il sistema deve garantire commenti per le sezioni chiave, per favorire la collaborazione della community Open Source	RNF_5
Categoria: Robustness			
2	DG_6 Robustezza agli errori causati dalla rete	Il sistema deve garantire un'elevata robustezza, gestendo qualsiasi errore causato dalla rete	RNF_10
Categoria: Deployment			
6	DG_7 Costi nulli per la persistenza dei	Il sistema per abbattere i costi relativi alla persistenza delle informazioni utilizza dei file	RNF_11

	dati		
Categoria: Portability			
7	DG_8 Portabilità su sistemi Linux-Debian	Il sistema deve permettere un'alta portabilità data dalla Virtual Machine di Python e dalla persistenza gestita tramite file	RNF_6

1.3 Design Trade-off

Trade-off	Descrizione
Tempi di risposta vs Memoria	Per migliorare i tempi di risposta del programma, tutti i dati persistenti saranno caricati in memoria già dall'avvio del software. Con conseguente aumento della Memoria occupata.
Costi vs Memoria	Per azzerare i costi relativi all'hosting dei server di storage, il software conterrà tutti i dati in file locali. A discapito della Memoria di massa occupata.

1.4 Definizioni, acronimi e abbreviazioni

Vengono riportate di seguito alcune definizioni presenti nel documento corrente:

- **Sottosistema:** un sottoinsieme dei servizi del dominio applicativo, formato da servizi legati da una relazione funzionale.
- **Design Goal:** le qualità sulle quali il sistema deve essere focalizzato.
- **Dati persistenti:** dati che sopravvivono all'esecuzione del programma che li ha creati e che dunque vengono salvati.
- **Mapping Hardware/Software:** studio della connessione fra parti fisiche e logiche di cui si compone il sistema
- **SDD:** System Design Document.
- **RAD:** Requirements Analysis Document.

1.5 Riferimenti

Di seguito una lista di riferimenti ad altri documenti utili durante la lettura:

- Requirements Analysis Document (RAD)
- System Design Document (SDD)
- Object Design Document (ODD)
- Test Plane (TP)
- Test Case Specification (TCS)
- Codice Sorgente
- Matrice di tracciabilità
- Il Documento segue le metodologie presentate nel libro: Object-Oriented Software Engineering, di Bernd Bruegge & Allen H. Dutoit

1.6 Organizzazione del Documento

Questo documento di System Design è composto da quattro sezioni:

Introduzione: Viene descritto in generale lo scopo del sistema, gli obiettivi di design che il sistema propone di raggiungere.

Architettura software corrente: Viene descritto lo stato attuale dell'architettura del software già presente.

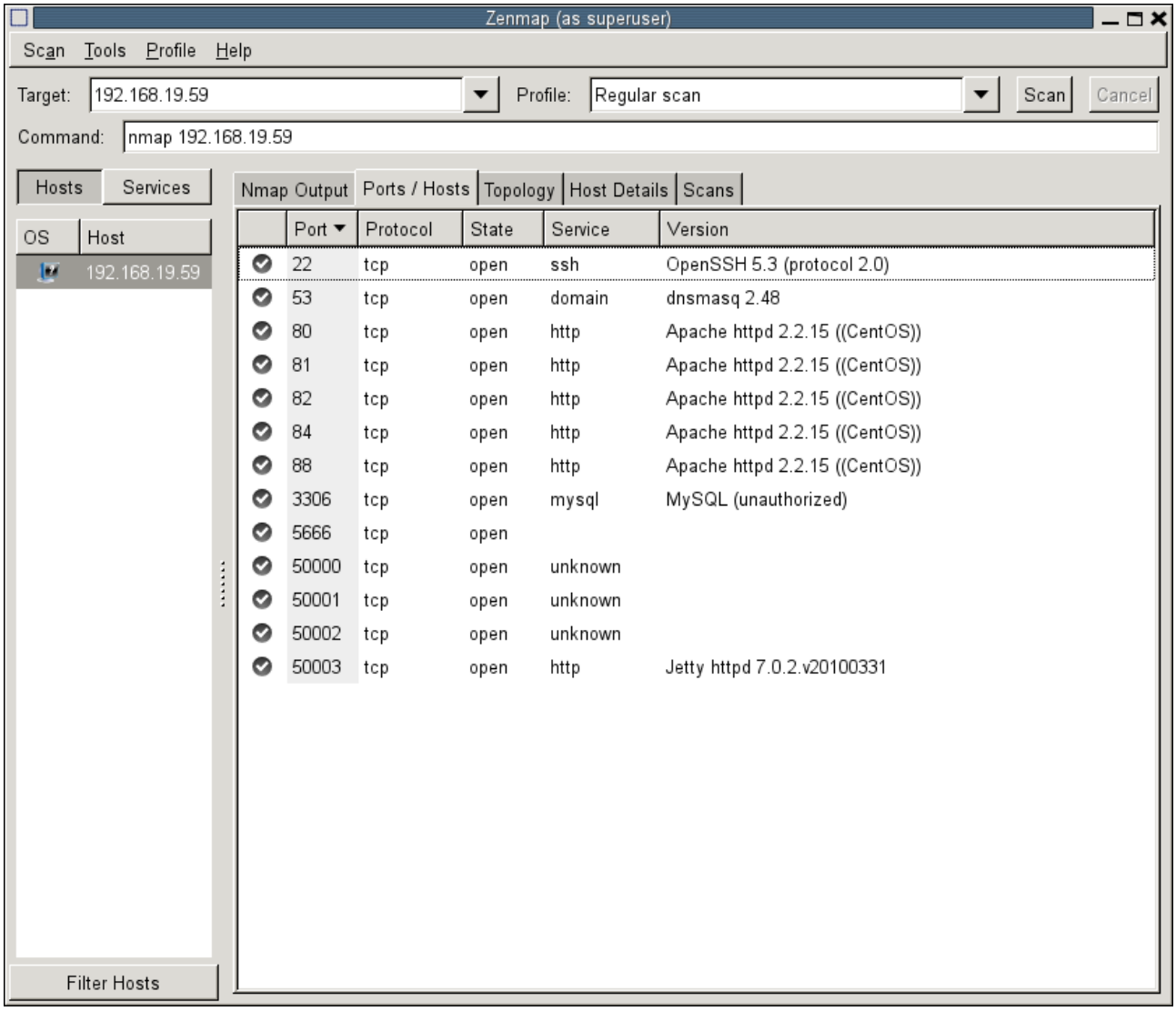
Architettura software proposta: Viene descritto come il sistema sarà definito e partizionato in sottosistemi, il loro mapping Hardware/Software, la gestione dei dati persistenti. Verranno poi presentate la struttura dei singoli sottosistemi e le boundary conditions riguardanti l'intero sistema.

Glossario: Contiene la lista dei termini usati nel documento con annessa spiegazione

2 Architettura del sistema corrente

Attualmente non esiste un software che abbia le stesse funzionalità volute in NetGun, essendo questo un progetto di Greenfield Engineering. Nonostante ciò, possiamo distinguere dei competitor che hanno funzionalità simili come Nessus, OpenVAS, Zenmap, etc…

ZenMap



3 Architettura del sistema proposto

3.1 Panoramica

Il sistema proposto è basato sullo stile architetturale Three Tier. Così da separare la logica di presentazione, di business e di accesso ai dati.

Lo stile architetturale Three Tier è stato scelto al fine di migliorare aspetti di qualità come:

- Modificabilità
- Leggibilità

Nello sviluppo del sistema sarà usato Python come linguaggio.

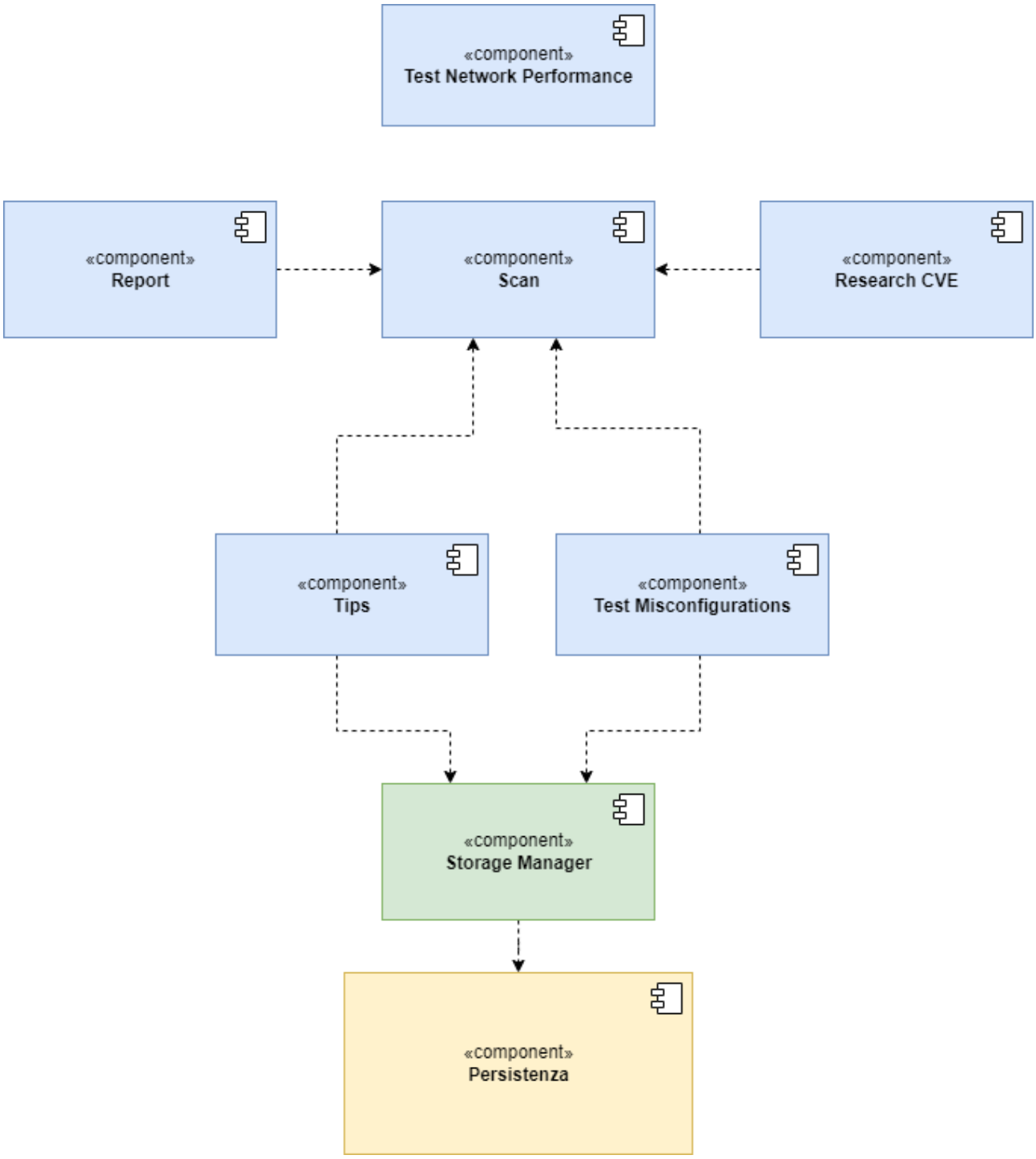
Per lo sviluppo della view sarà utilizzata la libreria Python Tkinter.

Per la gestione dei dati persistenti saranno usati file XML.

3.2 Decomposizione in sottosistemi

I sottosistemi individuati sono:

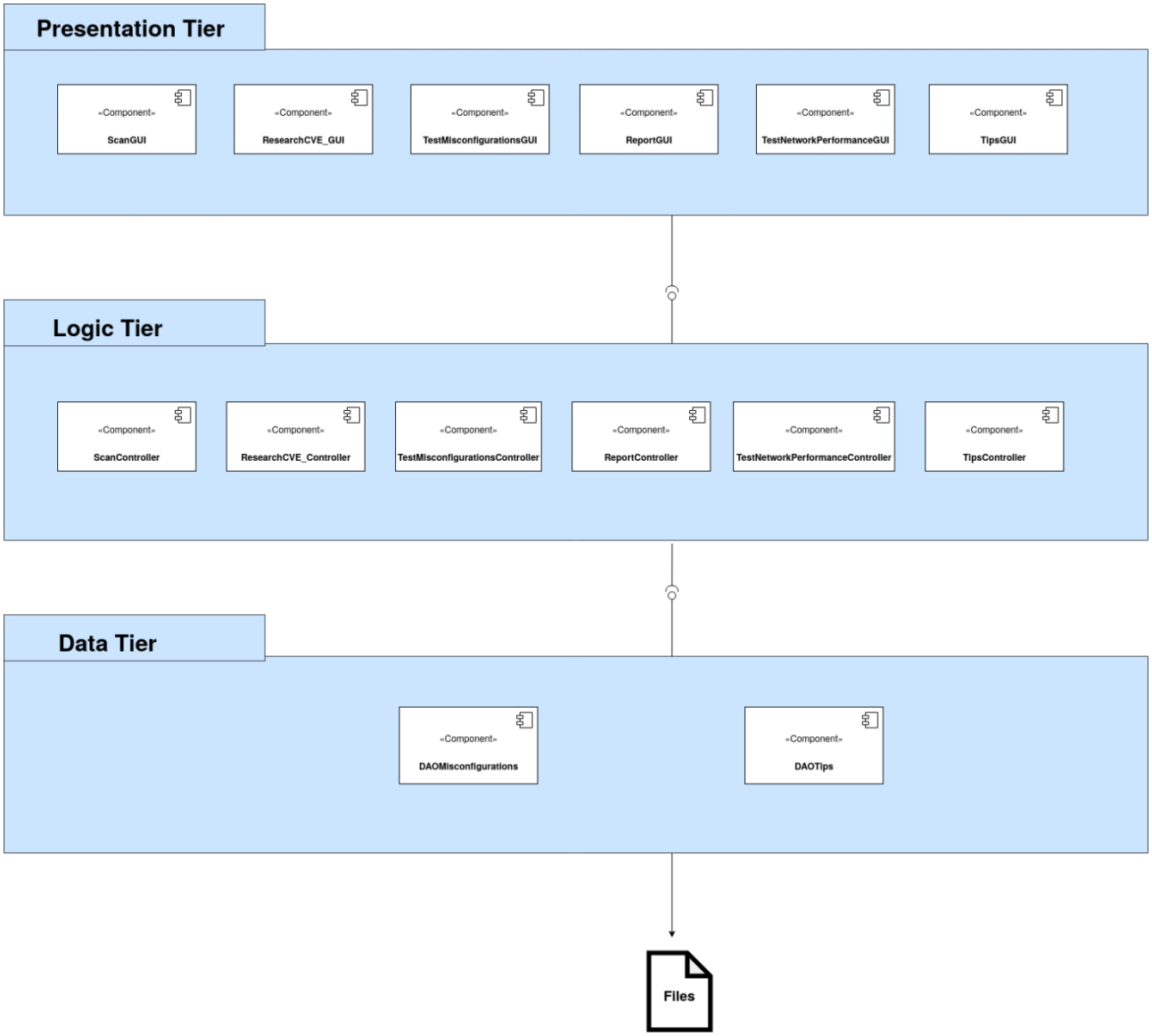
- **Scan**: si occupa di gestire lo scan da effettuare sul Target e i filtri con i quali configurarlo
- **Research CVE**: si occupa delle funzionalità di ricerca Common Vulnerabilities and Exposures sui servizi del Target
- **Test Misconfigurations:** si occupa di testare le mal configurazioni sui servizi del Target
- **Report**: si occupa delle funzionalità riguardanti l'elaborazione e l'esportazione di Report ottenuti dalle analisi effettuate
- **Test Network Performance**: si occupa di testare la stabilità della rete dell'utente
- **Tips**: si occupa di fornire all'utente dei consigli riguardo l'utilizzo dei servizi offerti dal Target
- **Persistenza**: Si occupa di gestire la persistenza dei dati tramite File
- **Storage Manager**: Si occupa di gestire i file di persistenza, la loro ottimizzazione, e l'interazione dell'intero sistema con questi


«component»
Test Network Performance
«component»
Report
«component»
Scan
«component»
Research CVE
«component»
Tips
«component»
Test Misconfigurations
«component»
Storage Manager
«component»
Persistenza

Component Diagram

Tramite questo Component Diagram è possibile comprendere la struttura Three Tier del sistema, distinguendo: Presentation Tier, Logic Tier e Data Tier, con i relativi sottosistemi che li compongono.

È possibile tramite la lollipop notation comprendere quali tier offrono un servizio e quali lo utilizzano.



3.3 Mapping Hardware/Software

Il Sistema che si desidera sviluppare, per essere utilizzato, necessità di una Python Virtual Machine eseguita su Sistemi Operativi Linux Debian-Based (Preferibilmente distribuzioni per il Penetration Testing).

La memorizzazione sarà gestita tramite file XML. Questi saranno Analizzati e Serializzati in Memoria di Massa dopo il primo avvio del sistema, consentendo così una riduzione significativa dei tempi di accesso ai dati persistenti.

Per poter usufruire della maggior parte delle funzionalità del sistema, è necessario che l'Hardware della macchina Host abbia una NIC (Network Interface Card) con tecnologia wireless o wired;

Il Sistema effettuerà Scansioni e Test di rete basandosi sullo standard TCP/IP per la comunicazione.

Run-Time Environment: Python VM

O.S.: Linux Debian-Based

Storage: XML Files and Serialized Files

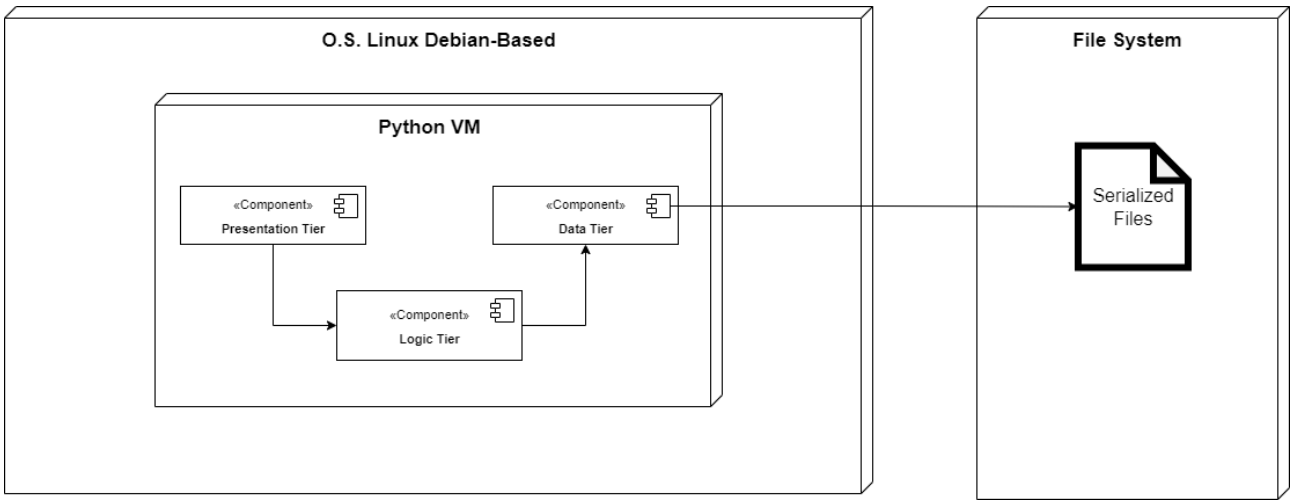
Hardware Components Required: NIC

Communications Standard: TCP/IP

Deployment Diagram

Il Software che sarà sviluppato si basa su una piattaforma Hardware avente un Sistema Operativo Linux Debian-Based con Macchina virtuale Python installata.

Il software dovrà essere eseguito nel Run-Time Environment della Python VM, e dovrà aver accesso al File System.



3.4 Gestione dei dati persistenti

Introduzione

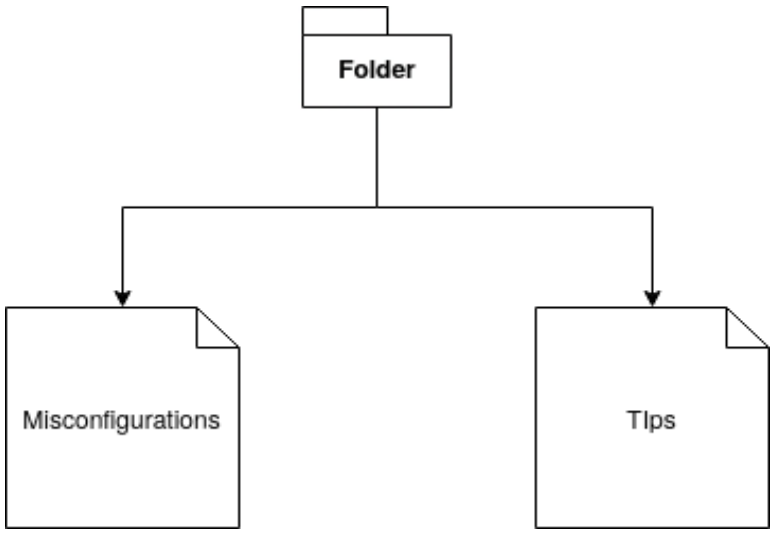
Per la gestione dei dati persistenti del sistema, si è deciso di utilizzare i file così da mantenere minimi i tempi di accesso alle informazioni.

La scelta di utilizzo dei file è stata presa al fine di mantenerci quanto più possibile coerenti con I design goals stabiliti, potendo contare su:

- **Costi nulli per la persistenza dei dati**, in quanto l'utilizzo di un DBMS avrebbe richiesto il mantenimento di un server, con conseguente aumento dei costi
- **Velocità di accesso ai dati persistenti**, poiché si intende utilizzare tecniche di ottimizzazione per diminuire il tempo di lettura medio dei file, come la serializzazione in memoria.
- **Modificabilità** poiché tramite file, si rende semplice la modifica da parte degli utenti anche dei dati persistenti, favorendo lo sviluppo e il miglioramento del sistema attraverso la community Open Source.

Le scelte sopra elencate hanno l'obiettivo principale di favorire: l'abbassamento dei costi di mantenimento delle informazioni, e lo sviluppo del software attraverso la community Open Source

Schema ad Albero descrivente la struttura dei file



Descrizione dei dati persistenti nei file:

- **Misconfigurations**: Contiene un insieme di mal configurazioni
- **Tips**: Contiene un insieme di consigli per l'utente

Formato dei dati persistenti

I dati saranno archiviati utilizzando il formato XML (eXtensible Markup Language), il quale permetterà una facile modifica da parte della community Open-Source.

3.5 Controllo globale del software

Il sistema NetGun richiede una continua interazione, quindi ogni funzionalità viene avviata dopo un'interazione o comando impartito dall'utente tramite l'uso dell'interfaccia grafica. Per questo motivo il sistema è di tipo Event-Driven Control.

Ogni volta quindi che l'utente avvierà un evento, quest'ultimo verrà gestito da un suo Handler, che indirizzerà l'intero controllo di flusso al sottosistema specifico per la funzionalità richiesta dall'utente.

3.6 Condizioni limite

Nel seguente paragrafo verranno presentate le boundary conditions inerenti a:

1. Avvio del sistema
2. Spegnimento del sistema
3. Fallimento del sistema

Avvio del sistema

Identificativo	UCBC_1 - Avvio del Sistema	Data	10/12/2022
		Versione	1.0
		Autori	Gianni
Descrizione	Lo UC permette l'avvio del sistema		
Attore principale	Utente		
Attori secondati	NA		
Entry condition	L'utente avvia il sistema		
Exit condition On success	Il sistema viene avviato correttamente		
Exit condition On failure	Il sistema non viene avviato correttamente		

Flusso di eventi principale		
1	Utente	Esegue sulla macchina il comando che avvia il sistema
2	Sistema	Verifica la sanità dei dati persistenti e rende disponibili le sue funzionalità all'utente e rende disponibili i dati, se sono sani.

Spegnimento del sistema

Identificativo	UCBC_2 - Spegnimento del Sistema	Data	10/12/2022
		Versione	1.0
		Autori	Gianni
Descrizione	Lo UC permette lo spegnimento del sistema		
Attore principale	Utente		
Attori secondati	NA		
Entry condition	L'utente spegne il sistema		
Exit condition On success	Il sistema viene spento correttamente		
Exit condition On failure	Il sistema non viene spento correttamente		

Flusso di eventi principale		
1	Utente	Esegue sulla macchina il comando di spegnimento del sistema
2	Sistema	Chiude tutti i processi e termina la sua esecuzione

Fallimento del sistema

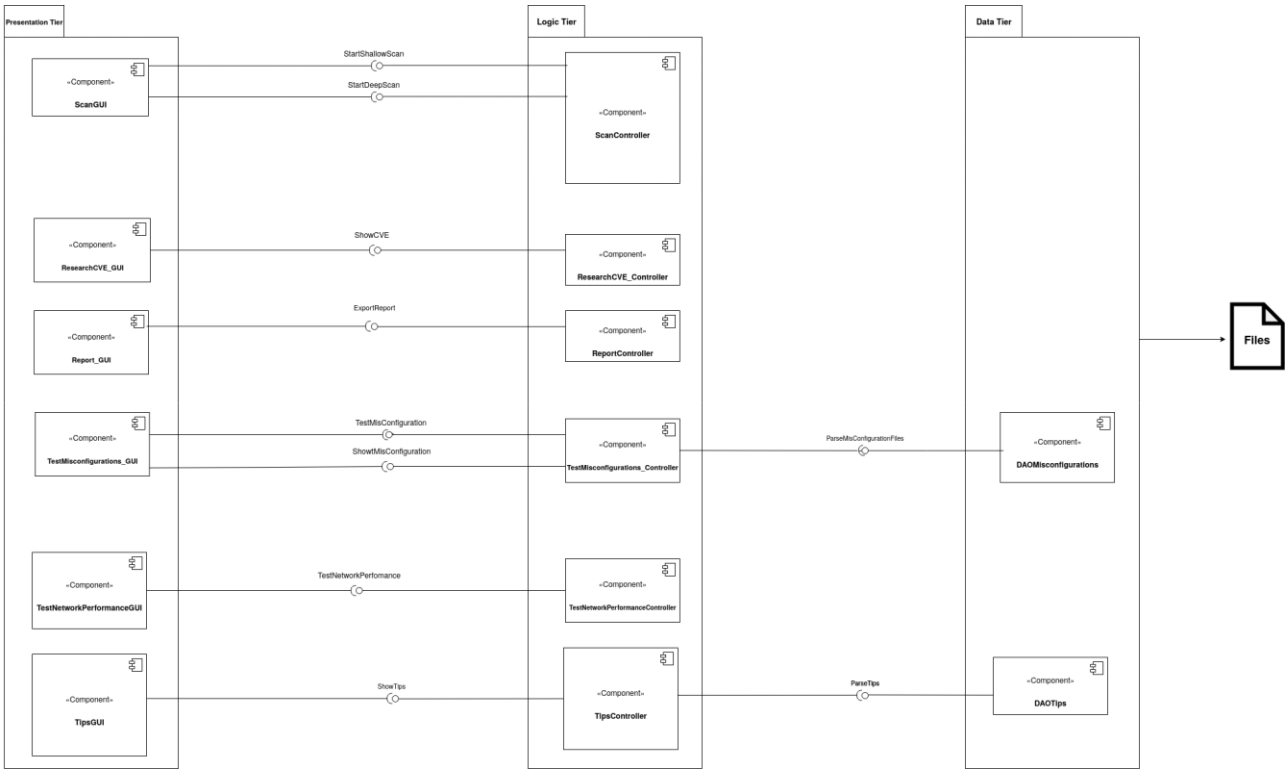
Identificativo	UCBC_3 - Fallimento del Sistema	Data	10/12/2022
		Versione	1.0
		Autori	Gianni
Descrizione	Lo UC definisce il comportamento del Sistema in caso di fallimento		
Attore principale	Utente		
Attori secondati	NA		
Entry condition	L'utente viene terminato inaspettatamente		
Exit condition On success	Il sistema viene riavviato correttamente		
Exit condition On failure	Il sistema non viene riavviato		

Flusso di eventi principale		
1	Utente	**Include UCBC_1**

Flusso di Eventi Alternativo: Il sistema non può avviarsi		
2.a1	Sistema	Invia un messaggio di errore per avvisare l'utente che il sistema non puo' avviarsi per un problema sconosciuto

4 Servizi dei Sottosistemi

In questa sezione verranno descritti i servizi di ogni sottosistema precedentemente elencati:



Servizi offerti dal Logic tier per il Presentation tier:

ScanController

Servizio	Descrizione
StartShallowScan	Questa funzionalità permette avviare una scansione non dettagliata
StartDeepScan	Questa funzionalità permette avviare una scansione dettagliata

ResearchCVE_Controller

Servizio	Descrizione
ShowCVE	Questa funzionalità permette di ricercare delle CVE per una data versione di un servizio, utilizzando il National Vulnerability Database (NVD)

Report_Controller

Servizio	Descrizione
ExportReport	Questa funzionalità permette di creare ed esportare un report

TestMisConfiguration_Controller

Servizio	Descrizione
TestMisConfiguration	Questa funzionalità permette di testare una malconfigurazione su un servizio
ShowMisConfiguration	Questa funzionalita' mostra le possibili misconfigurations di un servizio

TestNetworkPerfomanceController

Servizio	Descrizione
TestNetworkPerfomance	Questa funzionalità permette di testare le performance della rete che il sistema utilizzerà

TipsController

Servizio	Descrizione
ShowTips	Questa funzionalità permette di mostrare i tips consigliati per l'utente

Servizi offerti dal Data tier per il Logic tier:

DAOTestMisConfiguration

Servizio	Descrizione
ParseMissConfigurationFile	Questa funzionalità permette di convertire il formato XML del file delle miss configurations

DAOTipsController

Servizio	Descrizione
ParseTips	Questa funzionalità permette di convertire il formato XML del file dei tips