A DEVOPS-INTEGRATED CYBERSECURITY FRAMEWORK: UNIFIED THREAT INTELLIGENCE WITH AUTOMATED VULNERABILITY ASSESSMENT AND CONTINUOUS SECURITY TESTING

A PROJECT REPORT

Submitted By

PRANAV A N
[Reg. No. CH.EN.U4CYS21056]

RAGUL M
[Reg. No. CH.EN.U4CYS21062]

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

Under the guidance of

Dr. UDHAYA KUMAR S

Submitted to

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF COMPUTING
CHENNAI – 601103

April 2025

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956
SCHOOL OF COMPUTING CHENNAI

BONAFIDE CERTIFICATE

This is to Certify that this project report entitled **"A DEVOPS-INTEGRATED CYBERSECURITY FRAMEWORK: UNIFIED THREAT INTELLIGENCE WITH AUTOMATED VULNERABILITY ASSESSMENT AND CONTINUOUS SECURITY TESTING"** is the Bonafide work of **PRANAV A N [Reg. No. CH.EN.U4CYS21056] and RAGUL M [Reg. No. CH.ENU4CYS21062],** who carried out the project work under my supervision.

CHAIRPERSON SIGNATURE	**SUPERVISOR SIGNATURE**
Dr. Sountharrajan S	**Dr. Udhaya Kumar S**
Associate Professor	Associate Professor
Department of Computer Science and Engineering	Department of Computer Science and Engineering
Amrita School of Computing	Amrita School of Computing
Chennai.	Chennai.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE CANDIDATE

We declare that the report entitled **“A DEVOPS-INTEGRATED CYBERSECURITY FRAMEWORK: UNIFIED THREAT INTELLIGENCE WITH AUTOMATED VULNERABILITY ASSESSMENT AND CONTINUOUS SECURITY TESTING”** submitted by us for the degree of Bachelor of Technology is the record of the project work carried out by us under the guidance of **Dr. Udhaya Kumar S,** Associate Professor, Department of Computer Science and Engineering and this work has not formed the basis for the award of any degree, diploma, associateship, fellowship, titled in this or any other University or other similar institution of higher learning.

PRANAV A N
[Reg. No. CH.EN.U4CYS21056]

RAGUL M
[Reg. No. CH.EN.U4CYS21062]

ABSTRACT

In today's rapidly evolving cybersecurity landscape, organizations face continuous threats from sophisticated adversaries. Traditional security testing struggles to keep up with dynamic vulnerabilities, increasing risks and potential data breaches. This project presents a DevOps-integrated cybersecurity framework that enables Unified Threat Intelligence with Automated Vulnerability Assessment and Continuous Security Testing. Leveraging DevSecOps principles, it integrates security testing within the CI/CD pipeline, ensuring proactive threat identification throughout the software development lifecycle. The Unified Threat Intelligence tool developed in this project consists of three phases: Fingerprinting (Passive Reconnaissance), Active Reconnaissance, and Scanning & Enumeration. Unlike conventional tools, this framework integrates with GitHub Actions for automated security testing, performance benchmarking, and deployment automation, comparing execution efficiency, resource utilization, and scanning capabilities with industry-standard tools like Nmap, WhatWeb, Curl, SSLScan, and Dig. To enhance usability, the system features both a Command-Line Interface (CLI) and a Graphical User Interface (GUI) built using PyQt5. The CLI enables modular security testing for targeted scans, while the GUI simplifies interaction through intuitive module selection, real-time log visualization, and dynamic reporting. Additionally, Dockerized deployment ensures portability and scalability across computing environments, supporting cloud-native security automation. Performance analysis highlights its effectiveness in minimizing assessment time, optimizing CPU and memory usage, and reducing false positives, improving efficiency. CI/CD pipeline integration enforces security compliance and strengthens cyber resilience across software applications. By embedding continuous security assessment into DevOps workflows, this project bridges the gap between development, operations, and cybersecurity teams, ensuring proactive risk management. This framework represents a paradigm shift in cybersecurity, offering a scalable, automated, and intelligence-driven approach to threat mitigation. Future enhancements include machine learning for predictive threat analysis, cloud-based scalability, and real-time threat intelligence feeds, further fortifying security operations in an increasingly complex cyber threat landscape.

Keywords: DevSecOps, Vulnerability Assessment, Continuous Security Testing, CI/CD Pipeline, Threat Intelligence, Dockerized Deployment, Risk Management.

ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our honorable Chancellor **Sri Mata Amritanandamayi Devi,** for her blessings and for being a source ofinspiration. I am indebted to extend my gratitude to our Director, **Mr. I B Manikantan** Amrita School of Computing and Engineering, for facilitating us with all the facilities andextended support to gain valuable education and learning experience.

I register my special thanks to **Dr. V. Jayakumar**, Principal Amrita School of Computing and Engineering, **Dr. Sountharrajan**, Chairperson of Department of Computer Science and Engineering and **Dr. S. Udhaya Kumar**, Program Chair Department of CSE(CYS) for the support given to me in the successful conduct of this project and for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to the Project Coordinator, Review Panel Members and the entire faculty of the Department of Computer Science & Engineering, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

Pranav A N [Reg. No. CH.EN.U4CYS21056]

Ragul M [Reg. No. CH.EN.U4CYS21062]

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

VAPT	-	Vulnerability Assessment and Penetration Testing
OSINT	-	Open-Source Intelligence
RBAC	-	Role Based Access Control
CVSS	-	Common Vulnerability Scoring System
NVD	-	National Vulnerability Database
IoT	-	Internet of Things
CI/CD	-	Continuous Integration and Continuous Deployment
AWS	-	Amazon Web Services
GCP	-	Google Cloud Platform
SaaS	-	Software as a Service
DNS	-	Domain Name System
IP	-	Internet Protocol
TCP	-	Transmission Control Protocol
UDP	-	User Datagram Protocol
SSL	-	Secure Sockets Layer
TLS	-	Transport Layer Security
API	-	Application Programming Interface
Nmap	-	Network Mapper

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND STUDY

In today's digital age, cyber threats are becoming increasingly sophisticated, posing significant risks to businesses, governments, and individuals. The 2023 IBM Cost of a Data Breach Report revealed that the global average cost of a data breach has surged to $4.45 million, a 15% increase over the past three years. Recent cyberattacks, such as the 2023 MOVEit Transfer hack, which compromised millions of records from British Airways, the BBC, and the U.S. Department of Energy, highlight the growing need for proactive security measures. Additionally, the 2024 AT&T data breach, where 73 million customer records were leaked, underscores the severe consequences of failing to identify and mitigate security weaknesses. Organizations must adopt robust security frameworks to prevent such breaches, and Unified Threat Intelligence Framework plays a critical role in identifying and mitigating security flaws before they can be exploited.

Penetration testing has evolved significantly over the decades, beginning in the 1960s when the U.S. Department of Defense employed Tiger Teams to assess the security resilience of classified systems. By the early 2000s, organizations started leveraging automated security scanners to detect vulnerabilities in web applications and network infrastructures. Today, with the rise of DevSecOps, security testing is integrated into the software development lifecycle, ensuring continuous security monitoring. However, traditional vulnerability assessments often struggle to keep pace with rapid software updates, zero-day vulnerabilities, and AI-powered cyberattacks, necessitating the adoption of automated vulnerability assessment system which is capable of real-time security analysis and mitigation.

This project aims to develop an advanced Unified Threat Intelligence Framework tool that automates reconnaissance and vulnerability scanning, ensuring efficient detection of security risks. The tool is designed to execute three critical phases of penetration testing: Fingerprinting, which identifies system components and services; Active Reconnaissance, which gathers intelligence on potential attack vectors; and Scanning & Enumeration, which detects and categorizes security vulnerabilities. Furthermore, integrating security testing within DevOps workflows ensures vulnerabilities are assessed dynamically, reducing the risk of security gaps during frequent software deployments. With the increasing emphasis on cybersecurity compliance through regulations like the EU's NIS2 Directive and the U.S. SEC's cybersecurity disclosure rules, real-time security assessments are more crucial than ever.

To validate its effectiveness, this tool will be benchmarked against industry-standard security scanners, including Nmap, WhatWeb, Curl, and SSLScan, to compare execution time, CPU and memory utilization, and vulnerability detection accuracy. By automating and optimizing security assessments, this project addresses critical gaps in traditional penetration testing and enhances the efficiency, scalability, and accuracy of cybersecurity practices. With cyber threats continuously evolving, the importance of proactive security testing cannot be overstated, making this project an essential step toward fortifying digital infrastructures against modern cyberattacks.

1.2 PROBLEM IDENTIFICATION AND PROBLEM STATEMENT

As organizations increasingly rely on digital systems, the sophistication of cyber threats continues to grow, posing significant challenges to cybersecurity. Attackers exploit web application vulnerabilities, network weaknesses, and cloud misconfigurations, leading to data breaches, financial losses, and reputational damage. The 2024 Cybersecurity Threat Report states that over 60% of security incidents originate from unpatched vulnerabilities, highlighting the need for continuous security assessments. The surge in ransomware attacks, API vulnerabilities, and zero-day exploits exposes the limitations of traditional security testing methods, which often fail to detect threats in real time. Many organizations still rely on scheduled security audits, creating gaps that attackers exploit. Furthermore, the lack of DevOps integration and security automation hinders organizations from maintaining a proactive security posture.

Existing vulnerability assessment tools focus on network scanning and passive analysis, lacking comprehensive real-time risk assessment. Manual penetration testing, though effective, is time-consuming, resource-intensive, and prone to human error, making it unsuitable for modern agile development. Additionally, most security tools do not integrate into CI/CD pipelines, preventing continuous and automated security testing.

To address these challenges, this project proposes an automated Unified Threat Intelligence Framework that performs fingerprinting, active reconnaissance, and scanning & enumeration to provide real-time security insights. Unlike traditional tools, this solution integrates with GitHub Actions for continuous security validation within CI/CD pipelines. The framework is benchmarked against industry-standard tools like Nmap, WhatWeb, and SSLScan, evaluating execution time, resource utilization, and detection accuracy. By embedding security automation into the software development lifecycle, this project ensures early vulnerability detection and mitigation, significantly reducing cybersecurity risks.

1.3 GENERIC DIAGRAM

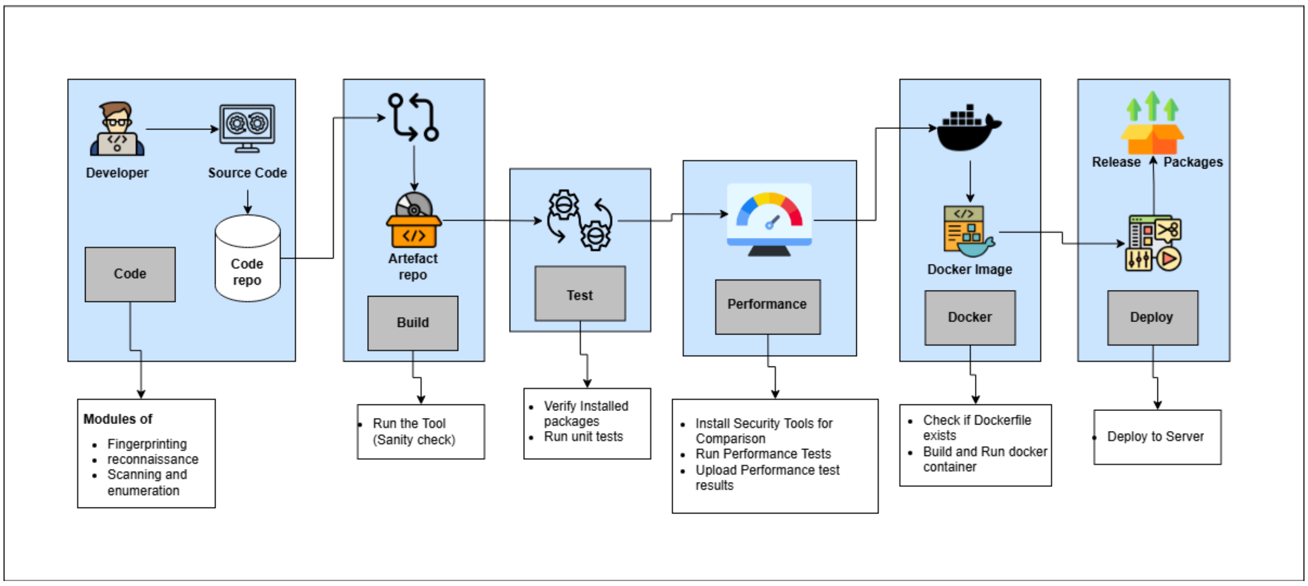


Fig 1.3: Generic Diagram represents Devops enabled Vulnerability Assessment System

The Fig 1.3 illustrates the general methodology for Vulnerability Assessment process and its seamless integration with DevOps workflows using CI/CD automation. It visually represents how security assessments are performed systematically and automated within the development lifecycle. The diagram consists of two major sections: the VAPT Framework and CI/CD Pipeline Integration.

This Framework section outlines the three primary phases of security testing: Fingerprinting (Passive Reconnaissance), Active Reconnaissance, and Scanning & Enumeration. Each phase includes specialized modules that facilitate intelligence gathering, vulnerability identification, and in-depth security assessments. The VAPT tool supports both Command-Line Interface (CLI) and Graphical User Interface (GUI) modes, ensuring flexibility for cybersecurity professionals and analysts.

The CI/CD Pipeline Integration section demonstrates how the VAPT tool is embedded into an automated security testing workflow using GitHub Actions. The pipeline includes key stages such as Build, Testing, Performance Benchmarking, Dockerization, and Deployment, ensuring continuous security validation and performance monitoring. Automated vulnerability assessments take place after every code commit, ensuring that security weaknesses are identified early in the development cycle.

By merging security automation with DevSecOps principles, the system enables real-time vulnerability detection, continuous monitoring, and proactive threat mitigation. This integration ensures that organizations can maintain a robust cybersecurity posture, reducing risks while maintaining the agility of modern software development practices.

CHAPTER 2

LITERATURE REVIEW

2.1 LITERATURE REVIEW BASED ON PREVIOUS RESEARCH PAPERS

The continuously evolving landscape of cybersecurity demands innovative solutions that integrate intelligence, automation, and collaboration. With the growth of attack surfaces and complexity in software systems, modern approaches have started focusing on proactive threat hunting, seamless DevSecOps integration, and human-centered security culture.

Gao et al.[1] in their study on Efficiently Hunting for Cyber Threats in Computer Systems Using Threat Intelligence, investigated the implementation of ThreatRaptor, a system designed to automate cyber threat hunting using Open-Source Cyber Threat Intelligence (OSCTI) and system audit logs. The authors examined how integrating OSCTI with audit logs enhances early-stage threat detection and reduces incident response time. Their findings demonstrated that ThreatRaptor effectively correlates system behaviors with threat intelligence feeds, minimizing false positives and improving detection accuracy. The authors concluded that this approach bridges the gap in real-time threat analysis, offering a proactive cybersecurity solution that outperforms traditional monitoring techniques.

Pelofske et al. [2] explored the application of graph theory in cyber threat detection in their study about Graph-Based Cyber Threat Intelligence for Threat Hunting. They implemented a Neo4j-based graph database to map Indicators of Compromise (IOCs) to Open-Source Intelligence (OSINT) sources, improving the visualization and correlation of attack patterns. The authors found that structuring cybersecurity data as a graph enhances the ability to detect hidden connections that conventional security tools might miss. Their study concluded that integrating graph-based approaches with threat intelligence allows security analysts to track evolving attack campaigns more efficiently and respond to threats proactively.

Bindlish et al.[3] investigated the automation of reconnaissance and scanning phases in penetration testing in their study on Automated Penetration Testing and Reconnaissance Using the RECON Tools. They introduced the RECON tool, a Bash-based script integrating security utilities such as Nmap, Nikto, and WPScan. The authors evaluated the efficiency of automated reconnaissance in identifying vulnerabilities in web applications and network infrastructures. Their findings revealed that automating reconnaissance improves accuracy and reduces the time required for vulnerability assessments. They emphasized that while automation enhances penetration testing efficiency, expert analysis remains crucial to interpreting results and mitigating false positives.

Kollepalli et al. [4] evaluated the effectiveness of automated security testing tools compared to manual penetration testing techniques in their study Evaluating Web Application Security Vulnerabilities: A Comparative Analysis of Automated and Manual Testing. They assessed various tools, including OWASP ZAP, Nikto, and Burp Suite, to detect web application vulnerabilities. The authors found that automated tools efficiently identify common vulnerabilities like SQL injection and cross-site scripting (XSS), but manual verification remains necessary for detecting complex security flaws. Their study concluded that a hybrid approach—combining automated scanning with manual testing—yields the most accurate security assessments while minimizing false positives.

Roy et al. [5] in their study about a Taxonomy of Adversarial Reconnaissance Techniques for Cybersecurity Threat Analysis, classified different reconnaissance methods used by attackers. They analysed third-party, system-based, and human-based techniques, providing a structured taxonomy of adversarial tactics. Their findings highlighted the need for organizations to implement countermeasures at various attack stages to mitigate reconnaissance-based threats. The authors concluded that understanding and classifying adversarial reconnaissance techniques enables security professionals to anticipate attacker methodologies and strengthen defensive strategies.

Abiona et al. [6] examined the role of DevSecOps in enhancing software security in their study on the Importance of DevSecOps Adoption for Secure Software Development. They investigated how integrating security automation within DevOps workflows minimizes vulnerabilities before they reach production. Their findings demonstrated that DevSecOps adoption reduces security risks by embedding security practices throughout the software development lifecycle. The authors concluded that organizations embracing DevSecOps experience improved security postures, faster threat detection, and enhanced compliance with industry regulations.

Ho-Dac et al. [7] explored the integration of open-source security tools in CI/CD pipelines in their study about Implementation of Open-Source Security Tools in CI/CD Pipelines. They focused on the incorporation of static application security testing (SAST) and dynamic application security testing (DAST) tools such as OWASP ZAP and SonarQube within Jenkins workflows. Their findings showed that automated security scanning helps detect vulnerabilities early in the development process, reducing human intervention while maintaining high detection accuracy. The authors concluded that leveraging open-source security tools in CI/CD pipelines enhances software security while ensuring continuous compliance with security best practices.

The author Thota et al. [8] studied the implementation of security automation in cloud-native DevSecOps environments in Cloud-Native DevSecOps: A Secure Software Development Lifecycle for Cloud Applications. The research explored the integration of security practices throughout the software development lifecycle by leveraging cloud-native technologies such as Kubernetes, container security tools, and Infrastructure-as-Code (IaC) scanning. Thota investigated how automating security testing and compliance enforcement within CI/CD pipelines enhances vulnerability detection and mitigates risks before deployment. The study also evaluated the effectiveness of shift-left security strategies, where security is embedded early in development rather than post-deployment. The findings highlighted that cloud-native DevSecOps significantly reduces security misconfigurations and improves software resilience against cyber threats, making it a critical approach for modern cloud-based applications.

Bolling et al. [9] studied the psychological aspects of security in DevOps in their study on DevOps Security Through an Applied Cyberpsychology Lens. They examined how human cognitive biases and stress levels impact security decision-making in software development environments. Their findings indicated that security misconfigurations often stem from psychological factors rather than technical limitations. The authors concluded that improving cybersecurity training programs and developing intuitive security tools can reduce errors caused by human factors, ultimately enhancing the overall security of DevOps practices.

Marandi et al. [10] investigated the automation of security scanning in CI/CD environments in their study on Automating Security Scanning in CI/CD Workflows: A Case Study Using GitHub Actions. They integrated security tools such as Snyk and StackHawk into GitHub Actions workflows to enable real-time vulnerability detection and automated alerts. Their findings revealed that continuous security scanning significantly reduces the time required to identify and remediate vulnerabilities. The authors concluded that integrating security automation within CI/CD pipelines strengthens an organization's security resilience and minimizes the likelihood of security breaches.

Cowell et al. [11] provided practical insights into automating DevOps workflows in their study about Automating DevOps Workflows with GitLab CI/CD: Best Practices and Security Implications. They demonstrated how organizations can embed security automation into GitLab CI/CD pipelines to improve software security. Their findings showed that automated security enforcement within DevOps workflows enhances threat detection while maintaining development efficiency. The authors concluded that DevOps teams adopting security automation practices experience reduced vulnerability exposure and improved software reliability.

The author Nagasundari S et al. [12] in their Extensive Review of Threat Models for

DevSecOps], analyzed various threat modeling approaches for DevSecOps environments. The study examined models such as STRIDE, DREAD, PASTA, and OCTAVE, assessing their effectiveness in securing CI/CD pipelines. The author highlighted challenges like automation, scalability, and integration with agile workflows. The research emphasized the need for adaptive threat modeling techniques that align with continuous security practices. The study concluded that AI-driven threat intelligence and automated vulnerability detection are essential for enhancing security in DevSecOps, ensuring proactive threat mitigation in evolving software development environments.

The author, Jin Yu Zhang et al. [13] in Quantitative DevSecOps Metrics for Cloud-Based Web Microservices investigated key performance indicators (KPIs) for assessing security in DevSecOps environments. The study introduced a comprehensive metric framework focusing on security vulnerability resolution time, compliance adherence, and security test coverage. By evaluating cloud-based microservices, the author demonstrated how these metrics provide insights into security effectiveness and operational efficiency. The findings emphasize the need for quantitative security assessments in DevSecOps to optimize automation and risk management. The research contributes to improving security posture by integrating measurable security controls within cloud-native application development.

The author Saima Rafi et al. [14] in their study about Prioritization-Based Taxonomy of DevOps Security Challenges Using PROMETHEE studied the security challenges in DevOps environments and proposed a prioritization framework using the PROMETHEE method. The study identified key security challenges faced in DevOps, including configuration management, access control, and compliance enforcement. The authors analyzed these challenges based on expert input and ranked them according to their severity and impact on security. The research provides a structured approach to addressing security concerns in DevOps pipelines, ensuring that organizations can implement risk-mitigation strategies effectively. The findings contribute to improving DevSecOps practices by highlighting critical vulnerabilities.

Markus Voggenreiter et al. [15] investigated automated security findings management in the context of industrial DevOps on Automated Security Findings Management: A Case Study in Industrial DevOps. The authors analyzed challenges in handling security vulnerabilities within continuous integration/continuous deployment (CI/CD) pipelines and proposed a framework to automate security findings management. Their study highlighted the inefficiencies of manual security reviews and emphasized the importance of integrating automated security triaging mechanisms. The proposed approach aimed to reduce alert fatigue, improve security response times, and enhance the overall efficiency of DevSecOps workflows.

The study concluded that automation significantly improves security posture in large-scale DevOps environments.

The author, Petar Lachkov et al. [16] in their study on Vulnerability Assessment for Applications Security Through Penetration Simulation and Testing, investigated penetration testing and vulnerability assessment techniques to enhance application security. The study highlighted the importance of automated security testing tools in identifying potential weaknesses in software applications. The author evaluated different penetration testing methodologies, emphasizing their role in mitigating risks and ensuring application resilience against cyber threats. The findings suggest that integrating penetration simulation into the DevSecOps pipeline can significantly improve security postures by enabling real-time detection and remediation of vulnerabilities. This research contributes to the field by demonstrating the necessity of continuous security assessment in modern software development.

The authors Nikolov et al. [17] proposed a framework for integrating automated threat modeling into Jenkins-based DevOps pipelines. Their approach systematically embeds security analysis within CI/CD workflows through four key phases: threat modeling using tools like OWASP Threat Dragon, structured data export, database integration for threat intelligence storage, and guided vulnerability scanning using SAST/DAST tools. The framework addresses critical DevSecOps challenges by enabling continuous security assessment without disrupting development velocity. The authors demonstrated how threat model outputs dynamically inform security scans, creating a closed-loop system where identified risks automatically trigger targeted testing. Their methodology represents a shift-left approach to security, ensuring vulnerabilities are detected and addressed earlier in the development lifecycle while maintaining DevOps efficiency through automation.

The authors Sermpezis et al. [18] investigated the integration of security into DevOps methodologies, emphasizing the transition to DevSecOps for enhanced software resilience. The authors analyzed risk areas in DevOps, including software development, application security, and infrastructure security, proposing mitigation strategies through Secure Software Development Framework (SSDF) practices. The authors highlighted the importance of tools like SAST, DAST, and container scanning in CI/CD pipelines to address vulnerabilities early. The authors also examined cultural and technical barriers to DevSecOps adoption, such as inter-team friction and inadequate tooling, while underscoring the benefits of automation and shared security responsibility in modern software development.

The authors Navya Priya et al. [19] proposed "FlawFix," an automated tool designed to scan, assess, and fix vulnerabilities in open-source library dependencies. The authors highlighted the growing reliance on open-source libraries in modern software development and the associated security risks. The authors compared existing vulnerability assessment tools like OWASP Dependency-Check and Snyk, noting their limitations in automated remediation. The authors emphasized FlawFix's unique capability to implement fixes automatically, reducing manual effort and errors. The authors validated the tool's effectiveness in small commercial applications, demonstrating a 70% reduction in vulnerabilities, while acknowledging future goals to expand compatibility with Gradle and npm projects.

The authors Shinde et al. [20] conducted a comprehensive study on Vulnerability Assessment and Penetration Testing (VAPT) as critical cybersecurity measures. Their research focused on analyzing prevalent web application vulnerabilities, particularly SQL Injection (SQLi) and Cross-Site Scripting (XSS), which ranked highly in the OWASP Top 10 list. The paper provided a detailed comparison between passive vulnerability assessment methods, which identify system flaws through scanning, and active penetration testing techniques that simulate real-world attacks to exploit vulnerabilities. The authors emphasized the complementary nature of these approaches when combined in VAPT methodologies. They demonstrated how organizations can leverage VAPT to proactively identify security weaknesses, assess risks, and implement appropriate countermeasures. The study also highlighted the growing importance of regular VAPT implementation due to increasing cyber threats targeting web applications. Additionally, the authors discussed various tools and techniques used in the VAPT process, offering practical insights for security professionals.

CHAPTER 3

METHODOLOGY

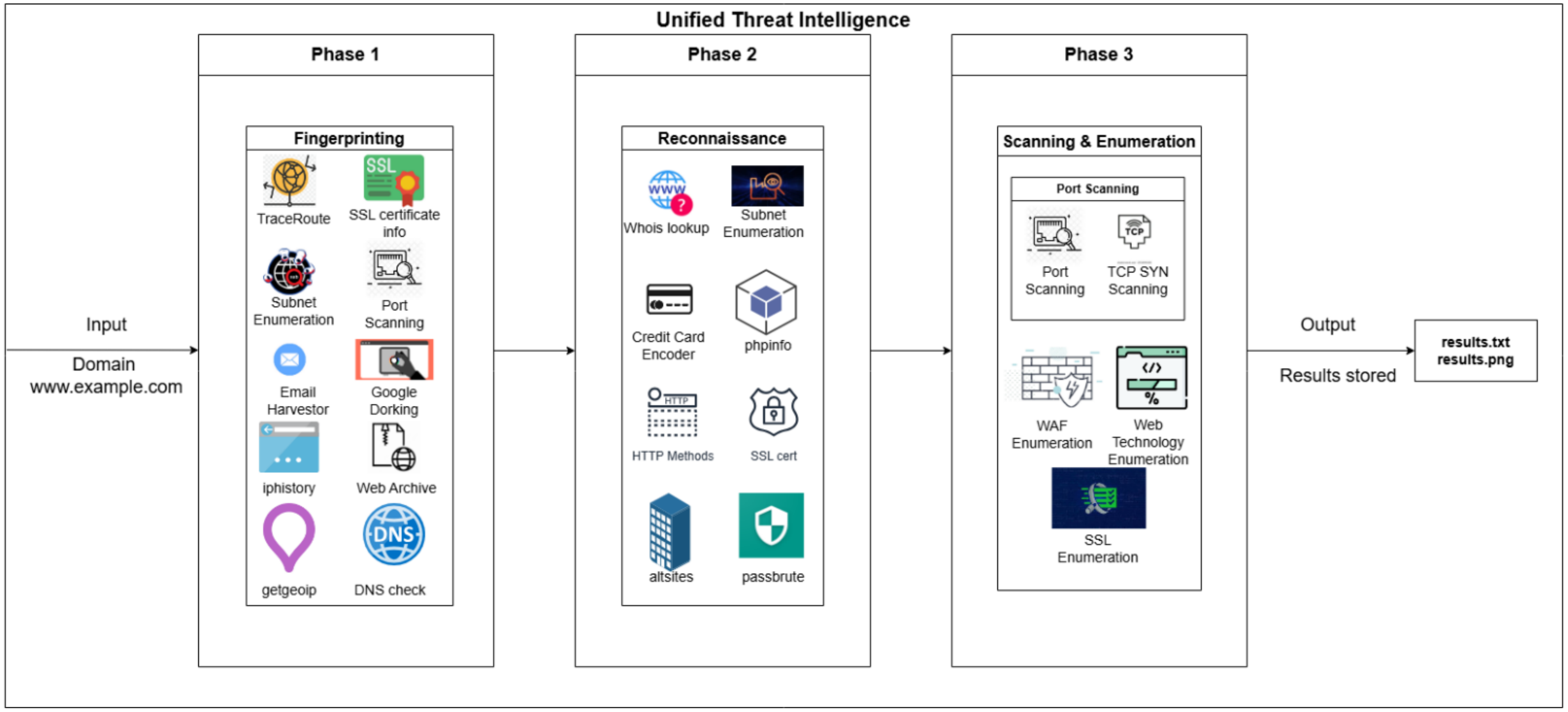


Fig 3.1: Methodology Flow Diagram

The methodology flow diagram Fig 3.1 visually represents the three-phase security assessment process of the Unified Threat Intelligence System, starting with an input domain (e.g., www.example.com) and progressing through Fingerprinting, Reconnaissance, and Scanning & Enumeration phases.

In Phase 1 (Fingerprinting), various modules such as Traceroute, SSL certificate info, Google Dorking, and Subnet Enumeration gather basic intelligence about the target. Phase 2 (Reconnaissance) deepens the analysis using Whois Lookup, HTTP Methods, and SSL certificate checks to identify potential vulnerabilities. Phase 3 (Scanning & Enumeration) performs port scanning, WAF enumeration, and web technology fingerprinting to detect security weaknesses.

The final output is stored as results.txt and results.png, ensuring structured documentation of security findings for further analysis.

3.1 OVERVIEW OF METHODOLOGY

The methodology adopted in this project follows a structured Vulnerability Assessment and Penetration Testing (VAPT) framework while incorporating DevOps automation to ensure continuous security assessment. The system is designed to automate vulnerability detection, improve security testing efficiency, and generate real-time performance benchmarks by comparing the tool’s effectiveness with other industry-standard security tools.

The VAPT methodology consists of three core phases that cover the entire reconnaissance and scanning process:

- **Fingerprinting (Passive Reconnaissance)** – This phase gathers publicly available information without actively interacting with the target, reducing the risk of detection.
- **Active Reconnaissance** – This phase involves direct interaction with the target system to enumerate open ports, services, and technologies used.
- **Scanning & Enumeration** – The final phase focuses on detailed scanning and identifying potential vulnerabilities in the system.

Additionally, DevOps integration is implemented using GitHub Actions-based CI/CD pipelines, enabling automated security testing and continuous benchmarking of the tool’s effectiveness.

3.2 STEPS INVOLVED IN VULNERABILITY ASSESSMENT PROCESS

This section outlines the structured approach adopted in the three phases of VAPT along with their corresponding security testing modules and functionalities.

3.2.1. Fingerprinting (Passive Reconnaissance)

The Fingerprinting Phase is the initial step in penetration testing. It involves passive reconnaissance techniques, which collect publicly available information about the target system without actively interacting with it. This phase is crucial for stealthy information gathering, helping security analysts to create a digital footprint of the target before moving to active testing.

Table 3.2.1: Module Names and Description Table for Phase 1

S.NO	Module Name	Description
1	**checkuser**	Identifies **active usernames** associated with the target system by analyzing public repositories, leaked credentials, and forums.
2	**dig**	Performs **DNS lookups** to extract records related to the domain, including **A, MX, NS, and CNAME records** to understand hosting infrastructure.
3	**dnscheck**	Analyzes **DNS configurations** to identify potential misconfigurations, subdomain takeovers, and exposed internal systems.
4	**getgeoip**	Extracts the **geolocation of an IP address**, helping to determine the physical location of servers.
5	**googlesearch**	Conducts **Google searches** using predefined queries to collect indexed data about the target.
6	**googledorker**	Utilizes **Google Dorking** techniques to find exposed **sensitive information**, such as login portals, misconfigured databases, and admin panels.
7	**iphistory**	Utilizes **Google Dorking** techniques to find exposed **sensitive information**, such as login portals, misconfigured databases, and admin panels.
8	**iphistory**	Tracks **historical IP address changes** for a given domain, helping in **tracking infrastructure movement** and identifying previous security flaws.
9	**revdns**	Performs **reverse DNS lookups** to map IP addresses back to domain names, potentially uncovering **hidden assets**.

10	**webarchive**	Checks **historical snapshots** of websites via the **Wayback Machine**, revealing previous versions of web applications and potential old vulnerabilities.

3.2.2. Phase 2: Active Reconnaissance

The Active Reconnaissance Phase involves direct interaction with the target to collect detailed technical insights, such as open ports, running services, and security misconfigurations. Unlike passive reconnaissance, this phase actively queries the target, increasing the chances of detection.

Table 3.2.2: Module Names and Description Table for Phase 2

S.NO	Module Name	Description
1	**altsites**	Identifies **alternative versions** of the target website, including **mirrored, staging, or backup versions** that might have weaker security.
2	**apachestat**	Extracts **Apache server status information**, which may expose internal requests, load statistics, and debugging information if improperly configured.
3	**backbrute**	Performs **brute-force attacks** on known login pages using commonly used usernames and passwords to identify weak credentials.
4	**cms**	Detects the **Content Management System (CMS)** used by the website, such as **WordPress, Joomla, or Drupal**, helping to determine possible vulnerabilities.
5	**dotbrute**	Attempts **brute-force discovery** of hidden files, **dotfiles**, and misconfigured directories that may contain sensitive information.
6	**filebrute**	Identifies **open ports** and running services to map the network attack surface.
7	**getports**	Utilizes **Google Dorking** techniques to find exposed **sensitive information**, such as login portals, misconfigured databases, and admin panels.

8	**httpmethods**	Enumerates **HTTP methods** allowed on the server, identifying potential misconfigurations such as **PUT and DELETE methods** that can be exploited.
9	**phpinfo**	Checks for **exposed PHP configuration files** (`phpinfo.php`), which can reveal **sensitive environment details**.
10	**sslcert**	Extracts **SSL certificate details**, identifying **expired certificates, weak encryption standards, and misconfigurations**.

3.2.3. Phase 3: Scanning and Enumeration

The Scanning & Enumeration Phase focuses on detecting vulnerabilities, weak configurations, and potential exploits within the system by conducting aggressive scanning techniques.

Table 3.2.3: Module Names and Description Table for Phase 3

S.NO	Module Name	Description
1	**arpscan**	Scans for **active devices on a local network** using **ARP requests**, helping in **network topology mapping**.
2	**bannergrab**	Extracts **service banners** from open ports to identify running **server versions** and possible vulnerabilities.
3	**getcencys**	Queries the **Censys database**, an internet-wide scanner, to collect **external security insights** about the target system.
4	**nmapmain**	Uses **Nmap scanning** techniques to detect **open ports, services, and potential exploits**.
5	**osdetect**	Determines the **operating system** of the target host based on network fingerprinting.

6	**waf**	Identifies **Web Application Firewalls (WAFs)**, which help in determining potential **bypass techniques**.
7	**webscan**	Scans for common **web vulnerabilities**, such as **SQL Injection, Cross-Site Scripting (XSS), and Local File Inclusion (LFI)**.
8	**webtech**	Detects **web technologies** and frameworks used in the target application, such as **Django, Flask, Angular, or ReactJS**.
9	**port scanners & web crawlers**	Identify **open ports, services, and indexed pages** to assess **attack vectors** and exposed sensitive information.

3.3 CI/CD PIPELINE STRUCTURE

3.3.1. Overview of Pipeline

The Continuous Integration and Continuous Deployment (CI/CD) pipeline is a crucial component of the system, ensuring automated security testing, performance benchmarking, and seamless deployment of the VAPT tool. It enhances efficiency, scalability, and consistency by automating the testing and deployment of security tools. The pipeline consists of five primary stages that define the core workflow, along with additional secondary stages that enhance the overall automation process.

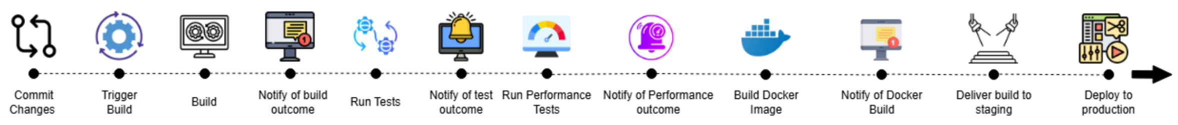


Fig 3.3: CI/CD Pipeline Structure

The pipeline consists of the following **five primary stages**, each serving a distinct purpose:

- **Build Stage** – The Build Stage is the first and foundational step of the pipeline. It is responsible for installing dependencies, setting up the runtime environment, and compiling the tool if required. This stage ensures that all necessary libraries, configurations, and required software components are available before proceeding to testing. Additionally, it validates that the system environment is correctly configured for execution. Without a successful build, the pipeline halts to prevent further errors down the workflow.

- **Testing Stage** – The Testing Stage is where the automated unit tests and security validation tests are executed to verify the correctness and functionality of the VAPT tool's security modules. These tests ensure that every module within the framework functions as expected and prevent potential misconfigurations. Integration tests validate the interaction between different modules, while performance and load tests verify system stability under varying workloads. Any failures at this stage trigger a notification, stopping further deployment until the issues are resolved.

- **Performance Benchmarking** – In this stage, the VAPT tool is compared against industry-standard tools like Nmap, WhatWeb, Curl, Dig, and SSLScan to evaluate execution efficiency. The primary metrics analyzed include execution time, CPU usage, memory consumption, and network overhead. The performance tests ensure that the VAPT tool remains optimized and competitive in terms of speed and resource utilization. These results are then automatically logged and visualized in graphs for further analysis.
- **Dockerization** – The Dockerization stage ensures that the VAPT tool is containerized using Docker, making it portable and easier to deploy across different environments. Docker creates a lightweight, self-sufficient container that includes all dependencies and eliminates compatibility issues. This process allows for faster deployments, consistency across different platforms, and easy scalability when running multiple security scans across various infrastructures.
- **Deployment Stage** – The final stage ensures that the latest validated and tested version of the VAPT tool is deployed in either a production or testing environment. The deployment step automates the release process, making sure every update is securely and efficiently made available to end users. The **deployment** process integrates with GitHub Actions and Docker, ensuring a seamless, scalable, and continuous deployment pipeline. Once the tool is deployed, security teams can access and utilize it for vulnerability assessment in real-world scenarios.

Apart from the five primary stages, the CI/CD pipeline incorporates additional secondary steps to enhance automation, monitoring, and communication within the workflow. These stages facilitate efficient execution and streamline the overall process:

- **Commit –** Developers push changes to the repository, triggering the pipeline.
- **Trigger Build –** Once code is committed, the pipeline automatically initiates the build process.

- **Notify of Build Outcome –** The system sends a notification upon successful or failed builds.
- **Run Tests –** Executes security tests to validate tool functionality.
- **Notify of Test Outcome –** Alerts developers about test results and any failed test cases.
- **Run Performance Tests –** Benchmarks execution time, CPU, and memory usage.
- **Notify of Performance Test Outcome –** Logs and reports performance evaluation.
- **Build Docker Image –** Creates a portable Docker container of the VAPT tool.
- **Notify of Docker Build –** Sends a notification once the container build is completed.
- **Deliver Build to Staging –** Deploys the tool in a staging environment for final validation.
- **Deploy to Production –** The validated build is automatically deployed for live use.

These additional steps ensure a smooth workflow, allowing teams to receive continuous feedback, monitor security changes, and maintain an efficient DevSecOps process. While these secondary stages provide additional automation and notifications, the five primary stages remain the backbone of the entire CI/CD pipeline, ensuring that security assessment and deployment are performed in a structured and reliable manner.

3.3.2. Steps involved in CI/CD Workflow

The CI/CD workflow consists of several automated processes designed to maintain the integrity, security, and efficiency of the tool. Below are the key steps involved:

1. Build & Test Stages:

These stages verify the correctness of the VAPT tool before it is tested for performance. The Checkpoints include:

- **Setup Job –** Initializes the runner and assigns required resources.
- **Checkout Repository –** Fetches the latest version of the VAPT tool from the GitHub repository.
- **Setup Python –** Configures the required Python version for execution.
- **Install Dependencies –** Installs required system packages and Python libraries.
- **Run VAPT Tool (Sanity Check) –** Executes a preliminary run to ensure the tool initializes correctly.
- **Post Setup & Post Checkout Repository –** Cleans up temporary files and logs if required.
- **Complete Job –** Finalizes the build and test process before proceeding to performance evaluation.

2. Performance Benchmarking Stage:

This stage measures execution time, CPU/memory usage, and network overhead while comparing the tool with industry-standard security scanners. The ckeckpoints include:

- **Setup Job –** Allocates resources for executing performance tests.
- **Checkout Repository –** Fetches the latest codebase.
- **Setup Python –** Ensures that the correct runtime environment is available.
- **Install Dependencies –** Installs the required system packages and libraries.
- **Install Security Tools for Comparison** – Installs Nmap, WhatWeb, Curl, Dig, SSLScan, theHarvester for benchmarking.
- **Run Performance Test –** Executes automated performance evaluation scripts to gather benchmarking results.
- **Upload Performance Test Results** – Stores test results in reports/performance_results.txt for further analysis.
- **Post Setup & Complete Job –** Finalizes the stage and prepares for Dockerization.

3. Dockerization Stage:

This stage **creates a Docker container** to package the tool for portability and scalability. The checkpoints include:

- **Setup Job –** Allocates necessary resources for Docker operations.
- **Checkout Repository –** Fetches the latest repository version.
- **Check If Dockerfile Exists –** Ensures a valid Dockerfile is present in the repository.
- **Run and Build Docker Container –** Builds and runs the Docker container for the VAPT tool.
- **Post Checkout & Complete Job –** Cleans up any temporary files and logs after successful containerization

4. Deployment Stage:

This stage **automates deployment** of the tool, ensuring it is ready for real-world usage. The checkpoints include:

- **Setup Job –** Assigns necessary resources for deployment.
- **Deploy to Server –** Executes deployment commands if deployment is required.
- **Post Checkout & Complete Job –** Finalizes deployment and cleans up any remaining artifacts.

CHAPTER 4

SYSTEM DESIGN AND EXPERIMENTAL WORK

4.1 SYSTEM REQUIREMENTS

4.1.1 Software Requirements

- **Operating System:** Ubuntu 22.04 / Windows Subsystem for Linux (WSL)
- **Programming Language:** Python 3.10+
- **Frameworks:** PyQt5 (for GUI), GitHub Actions (for DevOps integration)
- **Security Tools:** Nmap, Curl, WhatWeb, SSLScan

4.1.2 Hardware Requirements

- **Processor**: Intel i5 / AMD Ryzen 5 or higher
- **Memory:** 8GB RAM (for efficient scanning & benchmarking)
- **Storage:** Minimum 20GB free space

4.2 Simulation setup

4.2.1 Test Environment

To ensure accurate and repeatable testing, the simulation environment is configured to mimic real-world conditions while maintaining control over variables for precise performance evaluation. The test setup is as follows:

- **Deployed on:** The system is set up on VirtualBox running Ubuntu 22.04, which provides a stable and isolated environment for conducting security assessments without interference from external factors.
- **Target Domains Used:** To simulate real-world scanning scenarios, well-known test domains such as example.com and testsite.com are used. These domains are selected to evaluate tool accuracy and efficiency without violating ethical hacking guidelines.
- **Network Configurations:** The test environment is designed to include simulated public-facing IP addresses to replicate external scans. This setup allows assessment of network reconnaissance techniques while controlling external traffic influences.

The controlled environment ensures that benchmarking results for various security tools, including the VAPT framework, Nmap, WhatWeb, and SSLScan, remain consistent and comparable across multiple test runs.

4.2.2 Performance Testing

To evaluate the effectiveness of the Unified Threat Intelligence Framework (VAPT Tool), performance testing is conducted using key metrics that measure speed, resource utilization, and efficiency. The following performance aspects are monitored:

- **Execution Time Comparison:** The time taken by the VAPT tool to complete a security scan is compared against industry-standard tools such as Nmap, WhatWeb, Curl, Dig, and SSLScan. Faster execution time indicates a more efficient scanning mechanism.
- **CPU & Memory Utilization Monitoring:** The amount of CPU processing power and RAM consumption during scan execution is analyzed. Tools with lower CPU and memory footprints are preferred for resource-efficient environments.
- **Network Traffic Analysis:** The amount of data transmitted and received during scanning operations is evaluated to measure the impact of the tool on network bandwidth. This metric is crucial for determining the feasibility of deploying the tool in enterprise environments with network restrictions.

The results from these benchmarking tests are later visualized using graphical reports, helping users easily interpret tool performance and efficiency.

4.3 User Interface (GUI Version – PyQt5)

While the CLI-based VAPT tool offers flexibility for advanced users, a Graphical User Interface (GUI) version is developed using PyQt5 to enhance usability, accessibility, and visualization of results. The GUI allows users to interact with the Unified Threat Intelligence Framework in an intuitive way, eliminating the need for complex command-line operations. Users can effortlessly navigate through different modules, initiate security scans, and view real-time results using interactive visual components. This user-friendly approach makes the tool more accessible to security professionals who prefer graphical dashboards over terminal-based outputs.

4.3.1 Features of GUI-Based Unified Threat Intelligence Framework:

The **GUI version** of the tool offers the following features:

- **Module Selection:** Users can select specific fingerprinting, reconnaissance, or scanning modules from a list of available security tests. This customization feature allows users to focus on targeted assessments rather than running unnecessary scans.
- **Real-Time Execution:** The GUI dynamically updates logs, showing scan progress in real-time. Users can monitor each step of the security assessment as it happens, improving

transparency and control over the scanning process.

- **Graphical Reports:** The GUI automatically generates charts and graphs to represent security scan performance metrics. These visualizations help users easily interpret execution time, resource consumption, and detected vulnerabilities.

The GUI version also provides an export option, allowing users to save scan results and performance reports as images or PDFs. This feature is particularly useful for professionals who need to generate security reports for compliance and auditing purposes.

CHAPTER 5
RESULTS AND DISCUSSIONS

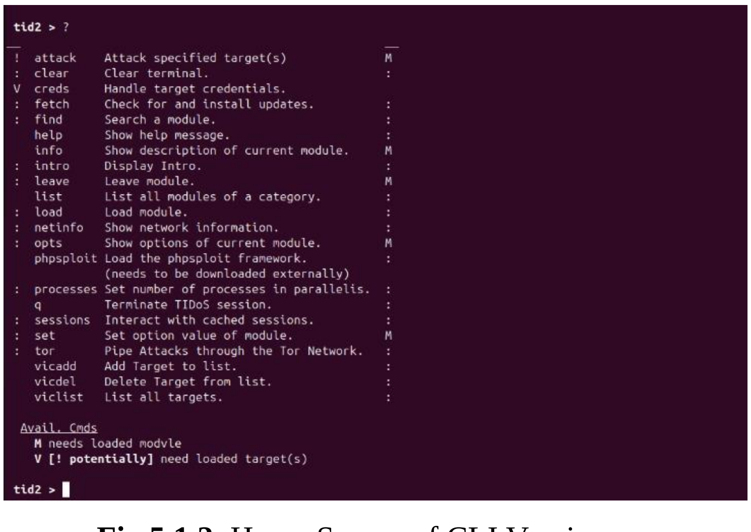
5.1 CLI VERSION OF THE UNIFIED THREAT INTELLIGENCE SYSTEM

5.1.1 Overview of CLI-Based Security Testing

The command-line interface (CLI) version of the VAPT tool provides a flexible and comprehensive way for security professionals to perform reconnaissance, scanning, and enumeration. The CLI-based approach is lightweight, faster, and scriptable, making it suitable for penetration testers who prefer automation and detailed console output.

5.1.2 Execution Flow Of CLI Version

- The user starts the tool using python3 tidconsole.py.
- The tool initializes and provides an interactive console interface where users can list available modules by typing help or ?.
- The user loads specific modules for reconnaissance, scanning, and enumeration.
- The tool performs the security test on the target domain and displays real-time output in the console.
- Results are logged for further analysis.



```
tid2 > ?
__                                                   __
!  attack    Attack specified target(s)              M
:  clear     Clear terminal.                         :
V  creds     Handle target credentials.
:  fetch     Check for and install updates.          :
:  find      Search a module.                        :
   help      Show help message.                      :
   info      Show description of current module.     M
:  intro     Display Intro.                          :
:  leave     Leave module.                           M
   list      List all modules of a category.         :
:  load      Load module.                            :
:  netinfo   Show network information.               :
:  opts      Show options of current module.         M
   phpsploit Load the phpsploit framework.           :
             (needs to be downloaded externally)
:  processes Set number of processes in parallelis.  :
   q         Terminate TIDoS session.                :
:  sessions  Interact with cached sessions.          :
:  set       Set option value of module.             M
:  tor       Pipe Attacks through the Tor Network.   :
   vicadd    Add Target to list.                     :
   vicdel    Delete Target from list.                :
   viclist   List all targets.                       :

 Avail. Cmds
   M needs loaded modvle
   V [! potentially] need loaded target(s)

tid2 > █
```

Fig 5.1.2: Home Screen of CLI Version

Figure 5.1.2 illustrates the output displayed when the user inputs "?" or "help" after loading the main module. It presents a list of available commands, including attack, clear, fetch, help, info, netinfo, and vicadd, among others. The vicadd command allows users to add a target URL (referred to as a victim), while also providing an option to display the list of previously added targets.

5.2 GUI VERSION OF THE UNIFIED THREAT INTELLIGENCE SYSTEM

5.2.1 Overview of GUI-Based Security Testing

The Graphical User Interface (GUI) version of the tool, developed using PyQt5, provides a user-friendly alternative to the CLI version. It allows users to interact with the tool using buttons, dropdowns, and visual indicators instead of command-line commands.

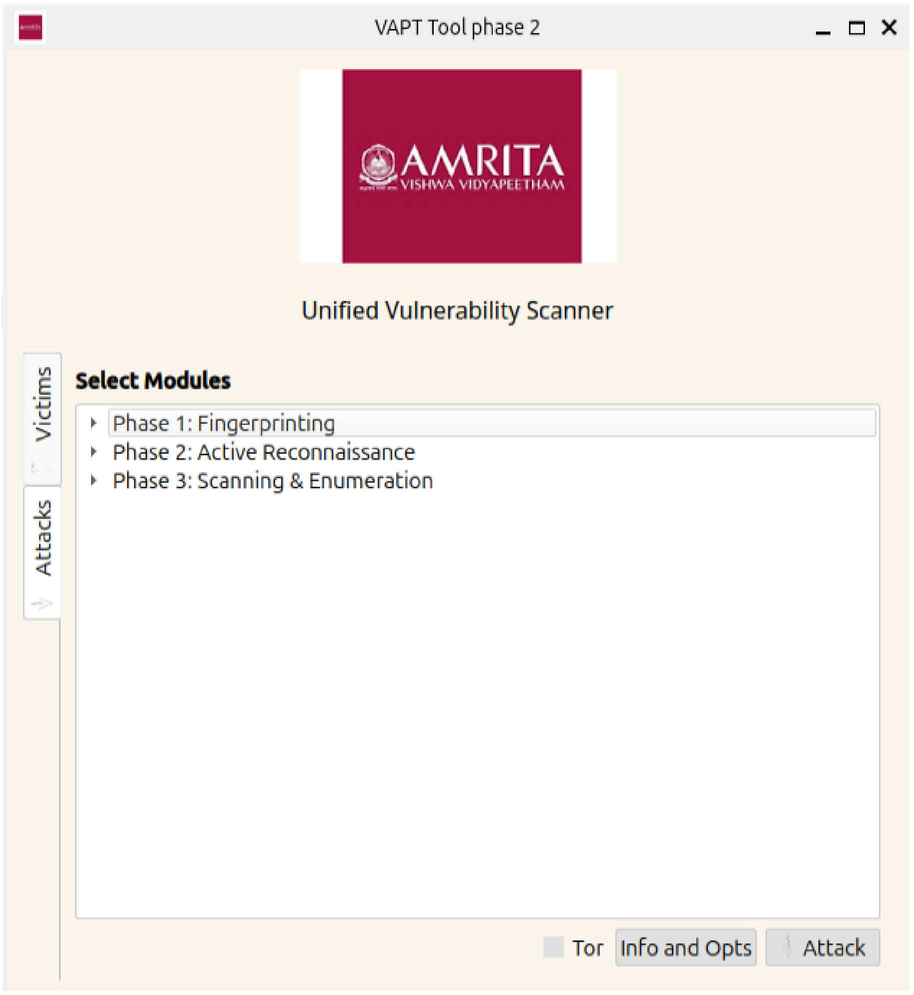
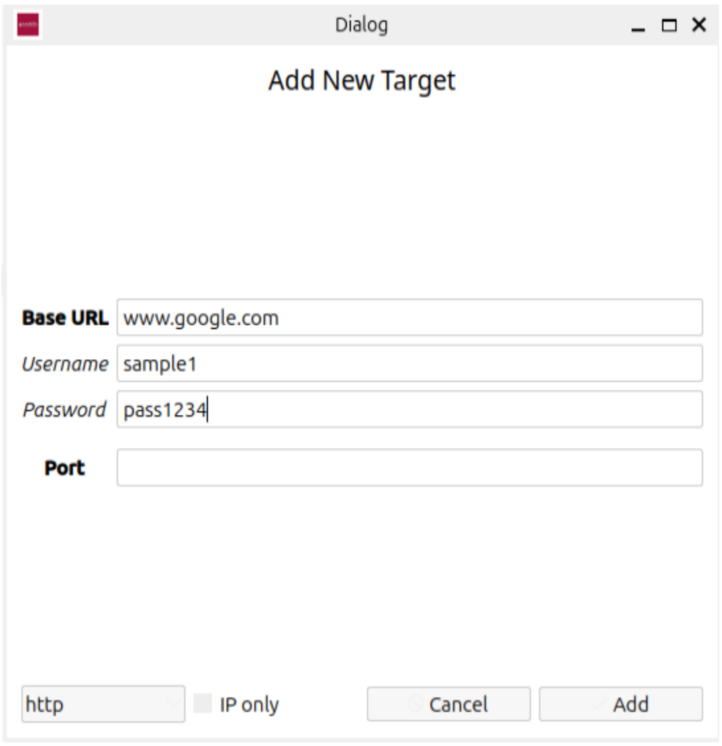

VAPT Tool phase 2
AMRITA VISHWA VIDYAPEETHAM
Unified Vulnerability Scanner
Victims
Attacks
Select Modules
Phase 1: Fingerprinting
Phase 2: Active Reconnaissance
Phase 3: Scanning & Enumeration
Tor
Info and Opts
Attack

Fig 5.2.1.1: Initial Screen of GUI Version

Figure 5.2.1.1 displays the home screen of the GUI, which presents all three phases of the Unified Threat Intelligence framework, from reconnaissance to scanning and enumeration. Additionally, it includes various options such as the Victims Bar for adding target domains, as well as Info and Opts buttons. These buttons provide essential information about input parameters, including the number of outputs, start year, and end year. The interface also features an Attack button, which initiates a vulnerability assessment once a victim is added.



Base URL: www.google.com
Username: sample1
Password: pass1234
Port:

Fig 5.2.1.2: GUI Screen with Input Dialog Box

The above figure Fig 5.2.1.2 displays the input dialog box, which allows users to configure essential parameters for the security assessment. It includes fields for entering the Base URL, Username, Password, and Port Number, along with options to specify HTTP, HTTPS, or IP-only mode. Additionally, the dialog box provides Save and Cancel buttons, enabling users to either confirm the input settings or exit without saving changes.

5.2.2 Execution Flow of GUI Version

- The user launches the GUI application, where the dashboard presents options to select security modules.
- The user selects the desired security phase (Fingerprinting, Active Reconnaissance, or Scanning & Enumeration) and inputs the target domain.
- The scan results are displayed dynamically in a log window, while performance metrics (CPU, execution time, memory usage) are plotted as graphs.
- The GUI allows users to export reports for documentation.

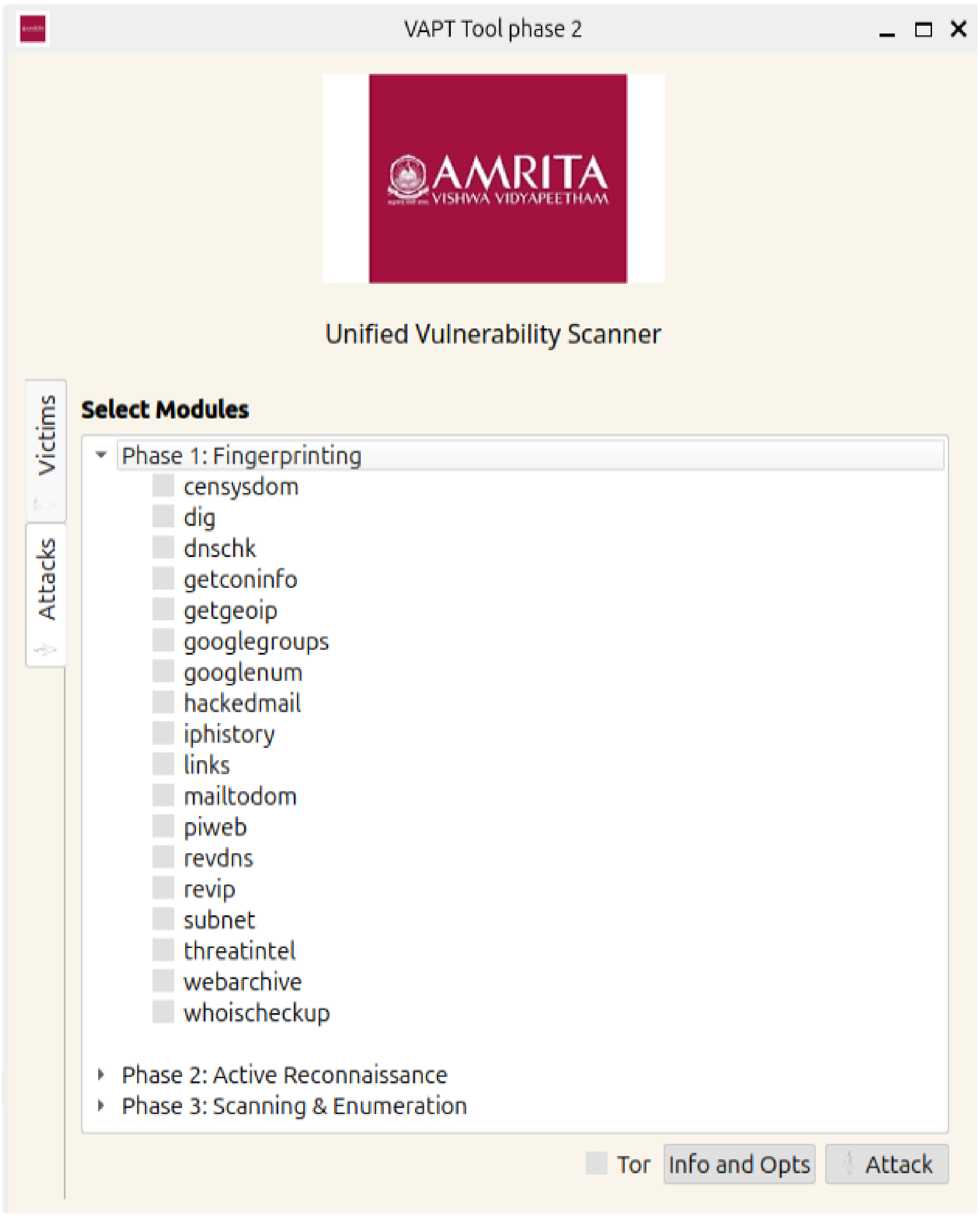


Fig 5.2.2: GUI Screen Displaying all available Modules

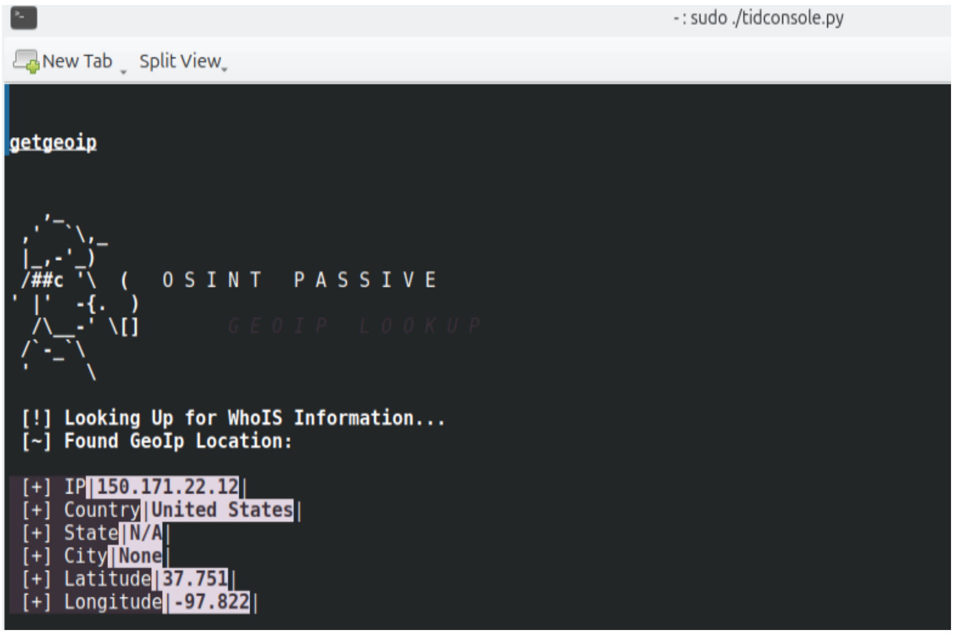
The above figure 5.2.2 illustrates the list of all available options under the Passive Reconnaissance section in the GUI version of the Unified Threat Intelligence Framework. This interface allows users to select modules simply by checking the corresponding checkboxes and clicking the "Attack" button to initiate the assessment. In contrast, the CLI version requires users to manually enter commands to load and execute each module. This makes the GUI version more user-friendly and accessible, especially for security analysts who prefer an interactive interface over command-line execution.

Similarly, for Active Reconnaissance and Scanning & Enumeration phases, the GUI provides an intuitive interface where users can select scanning modules such as port scanning, service detection, vulnerability enumeration, and web technology enumeration by checking the desired options.

This approach eliminates the need for manually typing commands, significantly improving ease of use and efficiency. The GUI ensures that all three phases of security assessment—Fingerprinting, Active Reconnaissance, and Scanning & Enumeration—can be performed seamlessly with minimal user intervention.

5.3 RESULTS OF UNIFIED THREAT INTELLIGENCE SYSTEM

5.3.1 RESULTS OF GUI VERSION



```
getgeoip


   ,'_
  ,'  `\,_
  |_,-'_)
 /##c '\  (   O S I N T   P A S S I V E
' |'   -{.  )
  /\__-' \[]        G E O I P   L O O K U P
 /`-_`\
 '     \

 [!] Looking Up for WhoIS Information...
 [~] Found GeoIp Location:

 [+] IP|150.171.22.12|
 [+] Country|United States|
 [+] State|N/A|
 [+] City|None|
 [+] Latitude|37.751|
 [+] Longitude|-97.822|
```

Fig 5.3.1: GUI Screen of Output

The above figure 5.3.1 displays the output of the "getgeoip" module in the GUI version of the Unified Threat Intelligence Framework. This module retrieves and presents key geographical details of a target, including IP address, country, state, city, latitude, and longitude. The structured format ensures that users can quickly interpret the results. While this figure specifically showcases the output of the "getgeoip" module, the results for other modules within the framework will be displayed in a similar structured manner, ensuring consistency and ease of analysis across different reconnaissance and scanning modules.

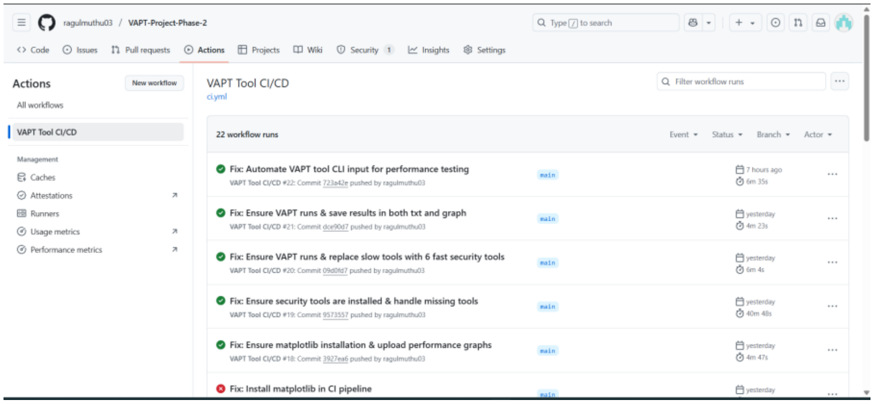
5.3.2 CI/CD PIPELINE EXECUTION RESULTS ON GITHUB ACTIONS

The CI/CD pipeline, implemented using GitHub Actions, was successfully executed.

The workflow includes:

- Automated security testing of VAPT tool modules.
- Performance benchmarking against industry-standard tools (Nmap, WhatWeb, Curl, etc.).
- Dockerization and deployment of the latest validated build.

The API developed in Python is used to facilitate communication between the frontend and the backend. External users, interacting via the React.js interface, submit smart contracts, and the Python backend processes the contracts, performs the analysis, and sends the results back to the frontend. The results summary is as follows:


ragulmuthu03 / VAPT-Project-Phase-2

Actions

VAPT Tool CI/CD
ci.yml

22 workflow runs

Fix: Automate VAPT tool CLI input for performance testing — VAPT Tool CI/CD #22: Commit 723a42e pushed by ragulmuthu03 — main — 7 hours ago — 6m 35s

Fix: Ensure VAPT runs & save results in both txt and graph — VAPT Tool CI/CD #21: Commit dce90d7 pushed by ragulmuthu03 — main — yesterday — 4m 23s

Fix: Ensure VAPT runs & replace slow tools with 6 fast security tools — VAPT Tool CI/CD #20: Commit 09d0fd7 pushed by ragulmuthu03 — main — yesterday — 6m 4s

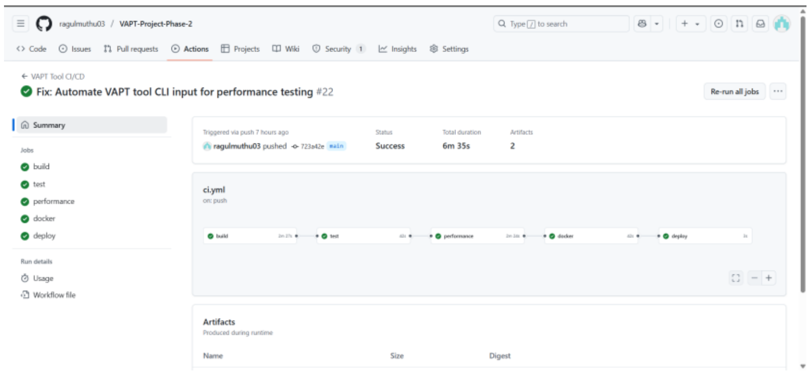
Fix: Ensure security tools are installed & handle missing tools — VAPT Tool CI/CD #19: Commit 9573557 pushed by ragulmuthu03 — main — yesterday — 40m 48s

Fix: Ensure matplotlib installation & upload performance graphs — VAPT Tool CI/CD #18: Commit 3927ea6 pushed by ragulmuthu03 — main — yesterday — 4m 47s

Fix: Install matplotlib in CI pipeline — main — yesterday

Fig 5.3.2.1: CI/CD Pipeline Execution Results

The figure 5.3.2.1 illustrates the successful creation of the CI/CD pipeline named "VAPT Tool CI/CD" within the project repository under the Actions section of GitHub. It provides details on the total number of workflow runs, which is displayed as 22 in this instance. Additionally, the figure highlights the status of each workflow execution, indicating whether the run was successful or failed, along with a timestamp and execution duration. The workflow is configured to operate within the main branch, ensuring that all updates pushed to this branch trigger the automated security testing and deployment pipeline. This visualization demonstrates the efficiency and reliability of the CI/CD integration, which plays a crucial role in maintaining the security and stability of the VAPT tool.


ragulmuthu03 / VAPT-Project-Phase-2

VAPT Tool CI/CD

Fix: Automate VAPT tool CLI input for performance testing #22

Triggered via push 7 hours ago — ragulmuthu03 pushed 723a42e main — Status: Success — Total duration: 6m 35s — Artifacts: 2

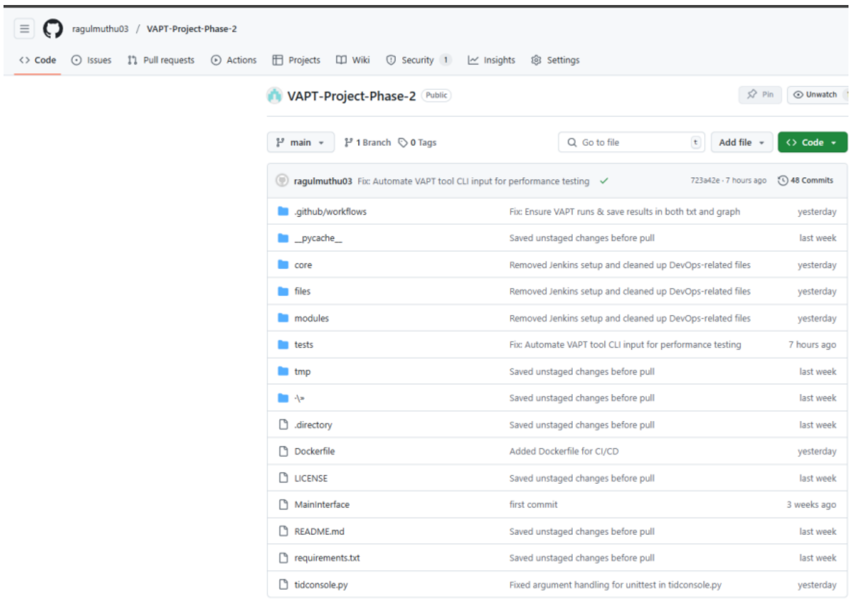
ci.yml
on: push

Artifacts
Produced during runtime

Name Size Digest

Fig 5.3.2.2: CI/CD Pipeline Flow

The above figure 5.3.2.2 illustrates the CI/CD pipeline flow, showcasing the sequential execution of various stages from Build to Deployment. It provides insights into the build status, indicating whether each stage—Build, Test, Performance Benchmarking, Dockerization, and Deployment—was executed successfully or encountered issues. Additionally, the figure displays the YAML configuration file used to define and automate the pipeline, ensuring consistency in execution. The total duration taken for the complete build process is also highlighted, demonstrating the efficiency of the automated workflow. This structured pipeline ensures seamless security validation, performance analysis, and deployment, integrating DevSecOps practices within the VAPT tool development lifecycle.



ragulmuthu03 Fix: Automate VAPT tool CLI input for performance testing ✓ 723a42e · 7 hours ago 48 Commits

.github/workflows	Fix: Ensure VAPT runs & save results in both txt and graph	yesterday
__pycache__	Saved unstaged changes before pull	last week
core	Removed Jenkins setup and cleaned up DevOps-related files	yesterday
files	Removed Jenkins setup and cleaned up DevOps-related files	yesterday
modules	Removed Jenkins setup and cleaned up DevOps-related files	yesterday
tests	Fix: Automate VAPT tool CLI input for performance testing	7 hours ago
tmp	Saved unstaged changes before pull	last week
·\»	Saved unstaged changes before pull	last week
.directory	Saved unstaged changes before pull	last week
Dockerfile	Added Dockerfile for CI/CD	yesterday
LICENSE	Saved unstaged changes before pull	last week
MainInterface	first commit	3 weeks ago
README.md	Saved unstaged changes before pull	last week
requirements.txt	Saved unstaged changes before pull	last week
tidconsole.py	Fixed argument handling for unittest in tidconsole.py	yesterday

Fig 5.3.2.3: GitHub Repository Structure

The figure 5.3.2.3 displays the GitHub repository structure containing all essential files required for the VAPT Tool project. It includes key directories and files such as workflow configurations, the core framework, various security testing modules, the Dockerfile for containerization, and the requirements file for dependency management. Additionally, the figure highlights the latest build triggered, indicating its success or failure with a corresponding tick or error icon. This provides a clear overview of the project's version control, automation pipeline, and integration of security testing workflows, ensuring smooth and efficient development within the CI/CD framework.

5.3.3 Evaluation Metrics

To assess the effectiveness of the Unified Threat Intelligence Framework, various evaluation metrics are considered. These metrics provide a quantitative and qualitative analysis of the tool's efficiency in comparison to industry-standard alternatives.

5.3.3.1 Execution Time:

This metric measures how long the tool takes to complete a security assessment. A lower execution time indicates a faster and more efficient scanning process. It is measured using Equation (1) in seconds by recording the time before and after the execution of each tool.

$$\text{Execution Time} = \text{End Time} - \text{Start Time} \qquad (1)$$

5.3.3.2 CPU Usage:

This factor evaluates the percentage of CPU resources utilized during the assessment. It is measured using Equation (2) and monitored using system utilities such as psutil to ensure that the tool runs efficiently without overloading system resources.

$$\text{CPU Usage (\%)} = (\text{CPU Time Used} / \text{Total Available CPU Time}) \times 100 \qquad (2)$$

5.3.3.3 Memory Consumption:

The amount of RAM used by the tool while running security assessments. This is measured using Equation (3) along with memory profiling tools to ensure that resource utilization is optimized for different computing environments.

$$\text{Memory Usage (\%)} = (\text{Memory Used} / \text{Total Available Memory}) \text{ x } 100 \qquad (3)$$

5.3.3.4 Network Resource Utilization:

This metric assesses the amount of network data transmitted and received during the scanning process. It is measured using Equation (4) and helps determine the bandwidth impact of running security tests on remote systems.

$$\text{Network Usage (MB)} = (\text{Data Sent} + \text{Data Received}) / 1024 \text{ x } 1024 \qquad (4)$$

5.3.4 Comparing Performance with other tools:

To evaluate the efficiency of the **VAPT tool**, a comparison was conducted across three key phases:

- Passive Reconnaissance (GeoIP Lookup)
- Active Reconnaissance (Reverse DNS Lookup)
- Scanning & Enumeration (Port Scanning)

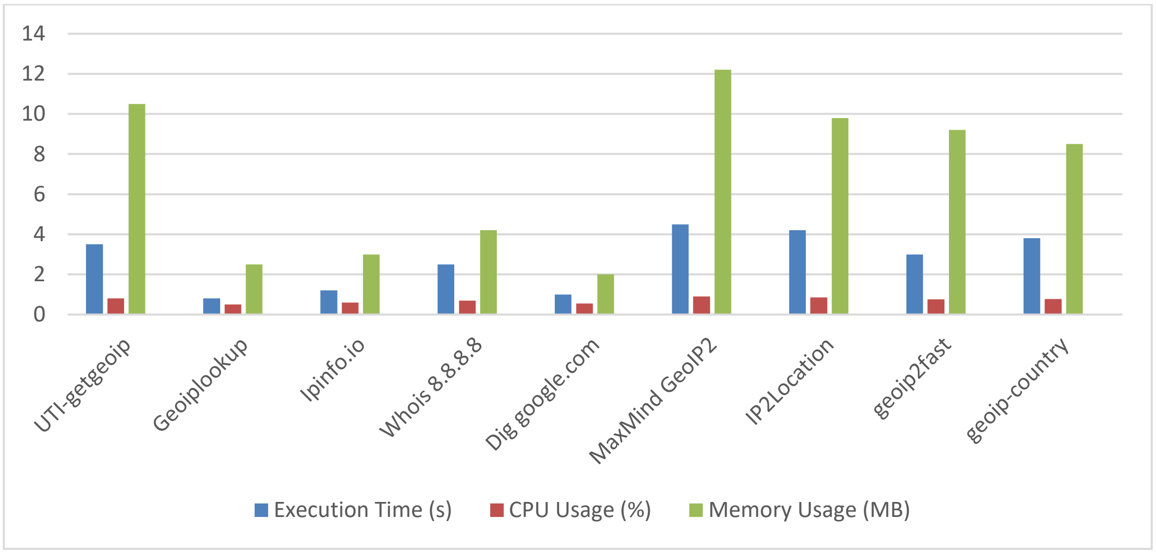
Each module of VAPT was benchmarked against popular open-source security tools in terms of:

- Execution Time (s): The time taken to complete the operation.
- CPU Usage (%): The percentage of CPU resources consumed.
- Memory Usage (MB): The RAM usage during execution.
- Accuracy: The correctness of results in comparison to known databases.

5.3.4.1 Passive Reconnaissance – GeoIP Lookup

The GeoIP Lookup module determines the geographic location of an IP address. The VAPT-GeoIP module was compared with well-known alternatives such as geoiplookup, ipinfo.io, whois, and dig.

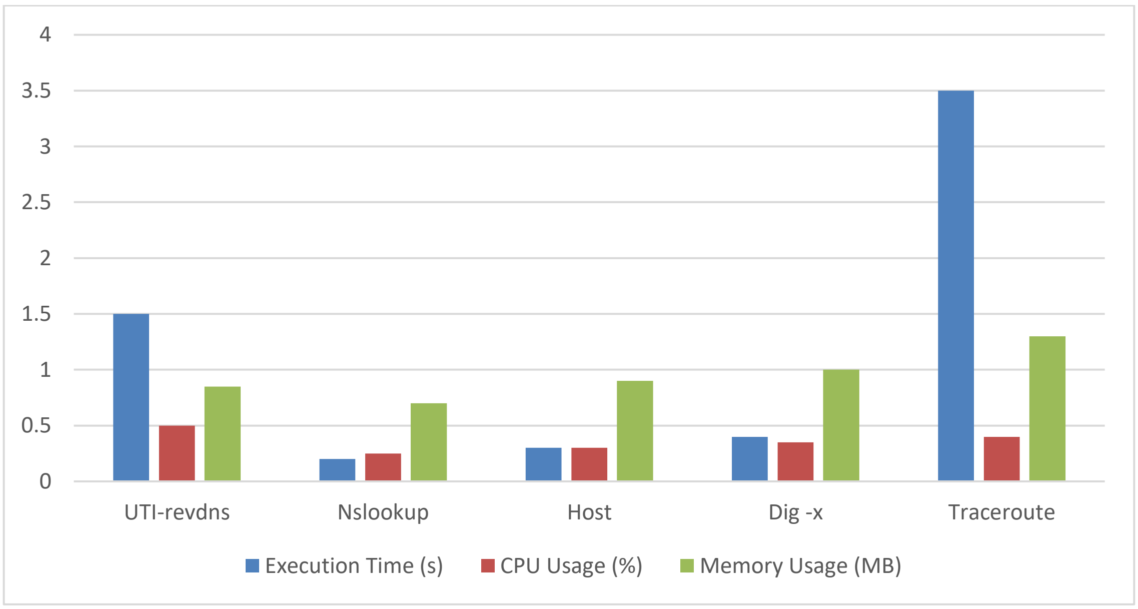
Figure 5.3.4.1 Performance Comparison of GeoIP Lookup Tools



5.3.4.2 Active Reconnaissance – Reverse DNS Lookup

Reverse DNS Lookup (revdns) identifies the hostname associated with an IP address. The VAPT module was compared with nslookup, host, dig -x, and traceroute.

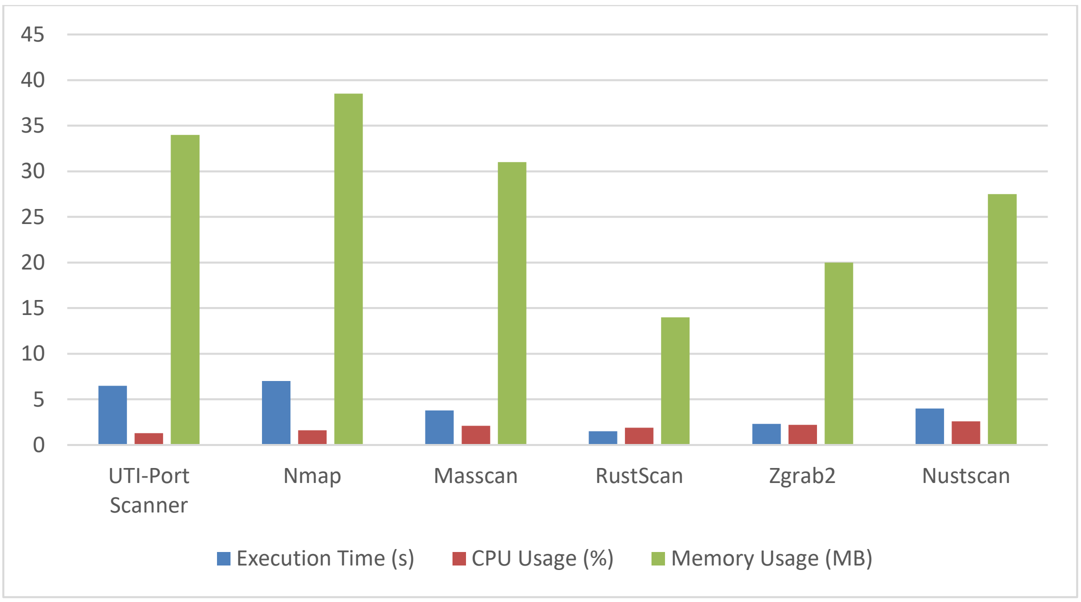
Figure 5.3.4.2 Performance Comparison of Reverse DNS Lookup Tools



5.3.4.3 Scanning & Enumeration - Port Scanning

Port scanning is a critical step in reconnaissance to identify open ports and services. VAPT's Port Scanner was compared with Nmap, masscan, rustscan, and zmap.

Table 5.3.4.3 Performance Comparison of Port Scanning Tools



CHAPTER 6

CONCLUSION AND SCOPE FOR FUTURE WORK

6.1 CONCLUSION

The Unified Threat Intelligence Framework (VAPT Tool) successfully integrates automated vulnerability assessment and penetration testing (VAPT) with modern DevOps practices using a GitHub Actions CI/CD pipeline. The project streamlines the process of fingerprinting, active reconnaissance, and scanning & enumeration, allowing security professionals to efficiently identify potential threats. The CLI version provides a command-driven approach to security testing, while the GUI version, developed with PyQt5, enhances the user experience through an interactive interface for module selection, scan execution, and results visualization. By automating security testing, the tool ensures that vulnerabilities are detected early in the development cycle, reducing risks and strengthening overall security posture. Additionally, performance benchmarking against industry-standard tools such as Nmap, WhatWeb, Curl, SSLScan, and Dig highlights its efficiency in execution time, CPU usage, and memory consumption.

The CI/CD pipeline enables continuous security assessment, ensuring seamless deployment and integration into existing security workflows. The project effectively automates reconnaissance and security scanning within a DevOps-integrated workflow while providing structured reporting and visualization for comprehensive vulnerability analysis.

Its efficiency is further validated through performance comparisons with well-established tools, demonstrating its capability for rapid and accurate security testing. By bridging the gap between automated security testing and DevOps integration, the VAPT tool makes vulnerability assessment an essential part of the software development lifecycle (SDLC), enhancing security at every stage of development.

6.2 SCOPE FOR FUTURE WORK

While the current implementation of the VAPT tool provides a comprehensive security testing framework, several enhancements can be introduced in the future to improve its capabilities and usability. Some key areas for future work include:

1. **Expanding Vulnerability Detection Modules:** Adding more scanning modules for API security testing, cloud security assessments, and IoT device penetration testing. Enhancing fingerprinting and reconnaissance with OSINT (Open-Source Intelligence) gathering techniques.

2. **Integration with Machine Learning for Threat Prediction**: Implementing AI-based anomaly detection to predict vulnerabilities and automate threat classification. Using machine learning models to analyze scan patterns and recommend mitigation strategies.

3. **Cloud-Based Deployment for Scalability:** Deploying the tool on cloud platforms like AWS, GCP, or Azure for large-scale security testing. Integrating serverless functions to run vulnerability assessments on demand without dedicated infrastructure.

4. **Multi-User and Role-Based Access Control (RBAC):** Developing a web-based dashboard with multi-user access, allowing organizations to collaborate on security assessments. Implementing role-based access control (RBAC) to restrict access based on user permissions.

REFERENCES

[1] P. Gao et al., "A System for Efficiently Hunting for Cyber Threats in Computer Systems Using Threat Intelligence," in 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 2021, pp. 2705-2708, doi: 10.1109/ICDE51399.2021.00309.

[2] E. Pelofske, L. M. Liebrock, and V. Urias, "Cybersecurity Threat Hunting and Vulnerability Analysis Using a Neo4j Graph Database of Open Source Intelligence," in Information, vol. 13, no. 8, 2022.

[3] Bindlish, S., Khurana, M., Chhabra, S., & Mahajan, S. (2021). RECON Tool: An Automation of Reconnaissance & Scanning. International Journal of Creative Research Thoughts (IJCRT), 9(8), 99–103. ISSN: 2320-2882.

[4] R. P. K. Kollepalli et al., "An Experimental Study on Detecting and Mitigating Vulnerabilities in Web Applications," in International Journal of Safety and Security Engineering, vol. 14, no. 2, 2024, pp. 523–532, doi: 10.18280/ijsse.140219.

[5] S. Roy et al., "Survey and Taxonomy of Adversarial Reconnaissance Techniques," in *ACM Computing Surveys*, 2022, doi: 10.48550/arXiv.2105.04749.

[6] Abiona, Oluwatosin & Oladapo, Oluwatayo & Modupe, Oluwole & Oyeniran, Oyekunle & Adewusi, Adebunmi & Komolafe, Abiola. (2024). The emergence and importance of DevSecOps: Integrating and reviewing security practices within the DevOps pipeline. World Journal of Advanced Engineering Technology and Sciences. 11. 127-133. 10.30574/wjaets.2024.11.2.0093.

[7] Ho-Dac, Hung & Vo, Van-Len. (2024). An Approach to Enhance CI/CD Pipeline with Open-Source Security Tools. European Modern Studies Journal. 8. 408-413. 10.59573/emsj.8(3).2024.30.

[8] R. C. Thota, "Cloud-Native DevSecOps: Integrating Security Automation into CI/CD Pipelines," International Journal of Innovative Research and Creative Technology (IJIRCT), vol. 10, no. 6, Dec. 2024, doi: 10.5281/zenodo.15036934.

[9] T. Bolling and R. G. Lennon, "Viewing DevOps Security Processes through An Applied Cyberpsychology Lens," in Proceedings of the 2023 Cyber Research Conference - Ireland (Cyber-RCI), pp. 97-103, doi: 10.1109/Cyber-RCI59474.2023.10671453.

[10] M. Marandi, A. Bertia, and S. Silas, "Implementing and Automating Security Scanning to a DevSecOps CI/CD Pipeline," in 2023 World Conference on Communication & Computing (WCONF), Raipur, India, pp. 1–6, doi: 10.1109/WCONF58270.2023.10235015.

[11] C. Cowell, N. Lotz, and C. Timberlake, Automating DevOps with GitLab CI/CD Pipelines: Build efficient CI/CD pipelines to verify, secure, and deploy your code using real-life examples, Packt Publishing, Feb. 2023. ISBN: 9781803233000.

[12] S. Nagasundari, P. Manja, P. Mathur and P. B. Honnavalli, "Extensive Review of Threat Models for DevSecOps," in IEEE Access, vol. 13, pp. 45252-45271, 2025, doi: 10.1109/ACCESS.2025.3547932

[13] J. Y. Zhang and Y. Zhang, "Quantitative DevSecOps Metrics for Cloud-Based Web Microservices," in IEEE Access, vol. 12, pp. 160317-160342, 2024, doi: 10.1109/ACCESS.2024.3486314.

[14] S. Rafi, W. Yu, M. A. Akbar, A. Alsanad and A. Gumaei, "Prioritization Based Taxonomy of DevOps Security Challenges Using PROMETHEE," in IEEE Access, vol. 8, pp. 105426-105446, 2020, doi: 10.1109/ACCESS.2020.2998819.

[15] M. Voggenreiter, F. Angermeir, F. Moyón, U. Schöpp and P. Bonvin, "Automated Security Findings Management: A Case Study in Industrial DevOps," 2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Lisbon, Portugal, 2024, pp. 312-322, doi: 10.1145/3639477.3639744.

[16] P. Lachkov, L. Tawalbeh and S. Bhatt, "Vulnerability Assessment for Applications Security Through Penetration Simulation and Testing," in Journal of Web Engineering, vol. 21, no. 7, pp. 2187-2208, October 2022, doi: 10.13052/jwe1540-9589.2178.

[17] L. Nikolov and A. Aleksieva-Petrova, "Framework for Integrating Threat Modeling into a DevOps Pipeline for Enhanced Software Development," 2024 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 2024, pp. 1-5, doi: 10.23919/SoftCOM62040.2024.10721871.

[18] E. Sermpezis, D. Karapiperis and C. Tjortjis, "Integration of Security in the DevOps Methodology," 2024 15th International Conference on Information, Intelligence, Systems & Applications (IISA), Chania Crete, Greece, 2024, pp. 1-6, doi: 10.1109/IISA62523.2024.10786669.

[19] N. P. N and V. P. H, "FlawFix:Automated Vulnerability Fixing Tool for Open Source Libraries," 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 1949-1953, doi: 10.1109/ICAC3N56670.2022.10074533.

[20] P. S. Shinde and S. B. Ardhapurkar, "Cyber security analysis using vulnerability assessment and penetration testing," 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave), Coimbatore, India, 2016, pp. 1-5, doi: 10.1109/STARTUP.2016.7583912.