 **HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION  
FACULTY OF INTERNATIONAL EDUCATION**------------------------------



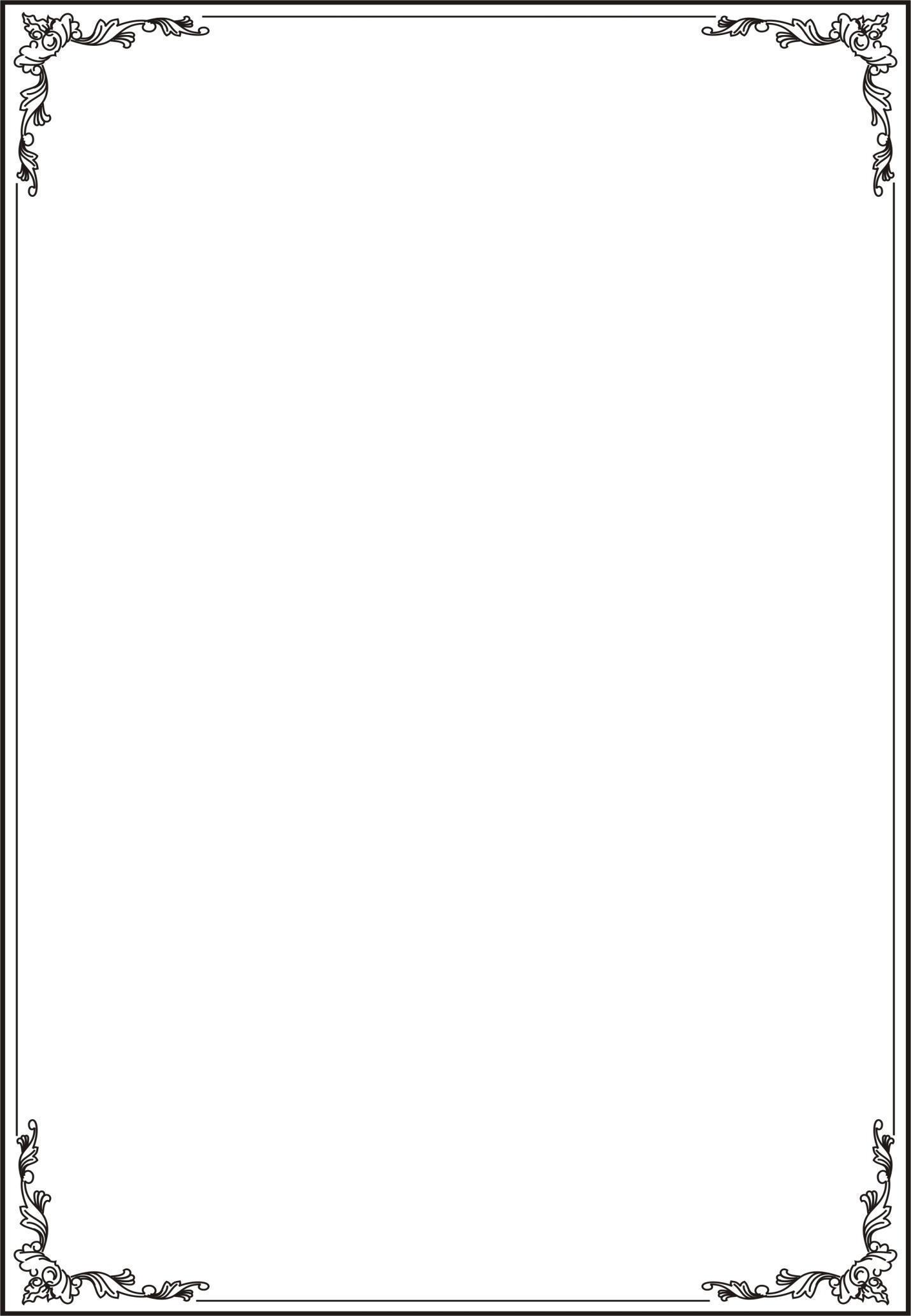
**PROJECT**

**DEVELOPING PENTEST AI AGENT**

**Instructor**: **Nguyễn Đăng Quang**

**Students: 21110780 - Nguyễn Duy Mạnh**

Ho Chi Minh City, Dec 2025

**HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION  
FACULTY OF INTERNATIONAL EDUCATION**------------------------------



**PROJECT**

**DEVELOPING PENTEST AI AGENT**

**Instructor**: **Nguyễn Đăng Quang**

**Students: 21110780 - Nguyễn Duy Mạnh**

Ho Chi Minh City, Dec 2025

**HCMC University of Technology and Education SOCIALIST REPUBLIC OF VIETNAM,**

**FACULTY OF INTERNATIONAL EDUCATION Independence – Freedom – Happiness**

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_**

**OBJECTIVES FOR PROJECT**

Student : Nguyen Duy Manh Student ID: 21110780

Major: Information Technology - Software Engineering

Project: Developing pentest ai agent

Instructor: Nguyễn Đăng Quang

1. **Objectives:**
2. **Time of execution**: 12 weeks

Signature of student:

*TP.HCM, Day…. month…. year….*

**HEAD OF IT DEPARTMENT** **INSTRUCTOR**

*(Sign with full name)*  *(Sign with full name)*

**INSTRUCTOR’S EVALUATION SHEET**

**Student Name**: Nguyễn Duy Mạnh **Student ID:** 21110780

**Major:** Software Engineering

**Project title:** Develop pentest ai agent

**Instructor:**

**EVALUATION**

**1.** Content of the project:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

**2.** Strengths:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

**3.** Weaknesses:

.......................................................................................................................................

**4.** Approval for oral defense? (Approved or denied)

.......................................................................................................................................

**5.** Overall evaluation: (Excellent, Good, Fair, Poor)

.......................................................................................................................................

**6.** Mark: ………. (in words: .........................................................................................)

Ho Chi Minh City, May 2025

**INSTRUCTOR**

*(Sign with full name)*

**REVIEWER’S EVALUATION SHEET**

**Student Name**: Nguyễn Duy Mạnh **Student ID:** 21110780

**Major:** Software Engineering

**Project title:** Develop pentest ai agent

**Instructor:**

**EVALUATION**

**1.** Content of the project:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

**2.** Strengths:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

**3.** Weaknesses:

.......................................................................................................................................

**4.** Approval for oral defense? (Approved or denied)

.......................................................................................................................................

**5.** Overall evaluation: (Excellent, Good, Fair, Poor)

.......................................................................................................................................

**6.** Mark: ………. (in words: .........................................................................................)

Ho Chi Minh City, May 2025

**REVIEWER**

*(Sign with full name)*

**ACKNOWLEDGEMENT**

*First and foremost, I would like to sincerely thank Mr. Nguyen Dang Quang for his steadfast support and guidance throughout the research and implementation of this project. His dedication and assistance in selecting the topic, refining methods, and addressing issues during the process have been invaluable.*

*We also wish to extend our heartfelt appreciation to the professors at Ho Chi Minh City University of Technology and Education, especially the Faculty Lecturers, for their enthusiastic teaching and for sharing their extensive knowledge and experience during my studies. Their contributions have been crucial to my successful completion of this project.*

*While we have strived for accuracy in our work, there may still be errors in the implementation. We are eager to receive constructive feedback from our instructors, as it will enable us to further enhance our skills and knowledge for future academic and professional endeavors.*

*Thank you sincerely!*

*Ho Chi Minh City, May 2025*

*Nguyễn Duy Mạnh*

**TABLE OF CONTENT**

[I. INTRODUCTION 1](#_Toc197605903)

[1.1 Reasons for choosing the topic 1](#_Toc197605904)

[1.2 Research objectives 2](#_Toc197605905)

[1.3 Research scope 2](#_Toc197605906)

[1.4 Project objectives 3](#_Toc197605907)

[1.5 Expected result 3](#_Toc197605908)

[II. THEORETICAL OF ARTIFICIAL INTELLIGENCE 4](#_Toc197605909)

[2.1 Overview of artificial intelligence 4](#_Toc197605910)

[2.1.1 Definition and formation of AI 4](#_Toc197605911)

[2.1.2 Major Development Paradigms in Artificial Intelligence 5](#_Toc197605912)

[2.1.3 Classification of AI by Capabilities 6](#_Toc197605913)

[2.1.4 Applications of Artificial Intelligence in Industry and Cybersecurity 8](#_Toc197605914)

[2.2 Machine Learning, Deep Learning, and the Path to LLMs and AI Agents 9](#_Toc197605915)

[2.2.1 Machine Learning – Traditional Machine Learning 9](#_Toc197605916)

[2.2.2 Deep Learning – Multilayer Neural Networks 10](#_Toc197605917)

[2.2.3 From Deep Learning to LLMs and AI Agents 11](#_Toc197605918)

[2.3 Generative AI 11](#_Toc197605919)

[2.3.1 Core Architectures in Generative AI 11](#_Toc197605920)

[2.3.2 Applications of Generative AI in Security Testing 12](#_Toc197605921)

[2.3.3 Impact on AI Agent Architecture and Role in AutoGen 12](#_Toc197605922)

[2.3.4 Limitations and Development Directions 13](#_Toc197605923)

[2.4 Agentic AI 13](#_Toc197605924)

[2.4.1 Definition 13](#_Toc197605925)

[2.4.2 How it work 15](#_Toc197605926)

[2.4.3 Benefit of Agentic AI 16](#_Toc197605927)

[2.4.4 Agentic AI and Generative AI 18](#_Toc197605928)

[2.5 AI Agent 19](#_Toc197605929)

[2.5.1 Definition 19](#_Toc197605930)

[2.5.2 How AI agents work 19](#_Toc197605931)

[2.5.3 Types of AI agents 20](#_Toc197605932)

[2.5.4 Popular Frameworks for Building AI Agents 20](#_Toc197605933)

[2.5.5 Single-Agent vs. Multi-Agent Systems 22](#_Toc197605934)

[2.5.6 Advantages of Multi-Agent Systems (MAS) 23](#_Toc197605935)

[III. IMPLEMENT AI AGENT SYSTEM FOR PENTEST 26](#_Toc197605936)

[3.1 Architectural Design and Framework Justification 26](#_Toc197605938)

[3.2 Pentest operations and processes 27](#_Toc197605939)

[3.3 Functions and roles of AI Agents 29](#_Toc197605940)

[3.4 Workflow of ai agent system 30](#_Toc197605941)

[IV. CODE IMPLEMENTATION 33](#_Toc197605942)

[4.1 System Architecture Overview 33](#_Toc197605946)

[4.2 Initialization and Directory Structure 33](#_Toc197605947)

[4.3 Agent Definitions 33](#_Toc197605948)

[4.3.1 Reconnaissance Agents 34](#_Toc197605949)

[4.3.2 Command Validation & Execution 35](#_Toc197605950)

[4.3.3 Vulnerability Scanning Agents 36](#_Toc197605951)

[4.3.4 Exploitation Agents 36](#_Toc197605952)

[4.3.5 Reporting and File I/O Agents 37](#_Toc197605953)

[4.4 Orchestration Using GroupChatManager 37](#_Toc197605954)

[4.5 Summary 38](#_Toc197605955)

[V. CONCLUSION 39](#_Toc197605956)

[5.1 Summary of Achievements 39](#_Toc197605958)

[5.2 Strengths and Limitations 39](#_Toc197605959)

[5.2.1 Strengths 39](#_Toc197605960)

[5.2.2 Limitations 39](#_Toc197605961)

[5.3 Future Development Plan 40](#_Toc197605962)

[VI. REFERENCES 41](#_Toc197605963)

# INTRODUCTION

## Reasons for choosing the topic

In today's digital age, cybersecurity threats have become increasingly sophisticated and prevalent, posing significant risks to organizations worldwide. According to the Cybersecurity Ventures Annual Cybercrime Report, cybercrime damages are predicted to cost the world $10.5 trillion annually by 2025[1]. Such alarming statistics highlight an urgent need for effective cybersecurity solutions, particularly automated penetration testing (pentest), which can proactively identify and address vulnerabilities before they are exploited by malicious actors.

Penetration testing, commonly known as ethical hacking, is a methodical approach to evaluating the security of information systems by simulating real-world cyberattacks. However, traditional manual pentesting methods are resource-intensive, time-consuming, and require substantial expertise. This creates a demand for integrating Artificial Intelligence (AI) to automate various phases of pentesting, including reconnaissance, vulnerability scanning, exploitation, and reporting.

The combination of AI with penetration testing addresses these challenges by significantly enhancing efficiency, accuracy, and consistency in vulnerability detection processes. Furthermore, this interdisciplinary topic aligns perfectly with my academic interests in cybersecurity and artificial intelligence, allowing me to apply theoretical knowledge practically to solve real-world problems.

Engaging in this project provides me the opportunity to acquire valuable technical skills, notably in advanced AI development and cybersecurity operations—both highly sought after in the contemporary job market. Experience gained through this project not only strengthens my professional portfolio but also prepares me effectively for future career paths.

Additionally, the field of cybersecurity is continually evolving, with automated security solutions becoming indispensable tools for organizations seeking robust protection against cyber threats. Developing an AI agent capable of effectively identifying vulnerabilities is not only intellectually stimulating but also contributes significantly to an emerging and impactful area of cybersecurity.

This project also bridges the gap between theoretical classroom education and practical industry applications. Hands-on experience with industry-standard tools and methods, including state-of-the-art frameworks such as AutoGen, enables me to understand real-world scenarios better, thereby providing a comprehensive foundation for future academic research and professional development.

## Research objectives

The primary objective of this research is to enhance understanding of penetration testing and explore the application of artificial intelligence (AI) in cybersecurity. This study aims to achieve the following:

Develop a simple AI agent model capable of performing basic penetration testing tasks such as vulnerability scanning, information gathering, and suggesting attack methods.

Gain practical experience in implementing machine learning algorithms for security applications, bridging the gap between theoretical knowledge from coursework and real-world cybersecurity challenges.

Evaluate the effectiveness of combining AI techniques with traditional penetration testing methodologies, identifying strengths and limitations of this approach.

Establish a foundation for potential future research and career development in the rapidly evolving field of AI-enhanced cybersecurity

## Research scope

This research focuses on developing a specialized AI agent system for penetration testing. Other cybersecurity fields such as malware analysis or digital forensics are not within the scope of this study. Specifically, the research scope includes:

Developing an AI agent system capable of performing basic tasks in the penetration testing process such as reconnaissance, vulnerability scanning, exploitation, and reporting.

Applying popular and effective pentesting tools such as Nmap, Gobuster, Nikto, Wapiti, and Nuclei to automate testing steps and enhance the capability of vulnerability detection.

Focusing on common and critical attack types such as SQL injection (SQLi), Cross-site scripting (XSS), Local File Inclusion (LFI), and Cross-site request forgery (CSRF).

Using a tightly controlled testing environment to ensure ethical standards and compliance with legal requirements related to cybersecurity.

Designing and implementing a multi-agent system using the AutoGen framework, in which specialized agents handle different stages of the testing process, thus achieving modularity and scalability of the system.

## Project objectives

The project aims to:

Develop an AI-driven penetration testing agent that can automate certain aspects of ethical hacking.

Integrate AI with traditional penetration testing tools to enhance efficiency and effectiveness.

Implement a cooperative multi-agent system to handle different testing phases, ensuring modularity and scalability.

Evaluate the performance of AI-based penetration testing techniques compared to traditional manual methods.

## Expected result

Through the development and testing of the AI penetration testing agent, the project is expected to:

Provide insights into how AI can enhance penetration testing methodologies.

Demonstrate the feasibility of using AI-driven automation in cybersecurity testing

Highlight the advantages and challenges of implementing AI in ethical hacking.

Serve as a stepping stone for further research in AI-powered security testing solutions.

This research will contribute valuable knowledge to both academic and practical cybersecurity fields, offering a starting point for future advancements in AI-driven penetration testing.

# THEORETICAL OF ARTIFICIAL INTELLIGENCE

## Overview of artificial intelligence

### Definition and formation of AI

Artificial Intelligence (AI) is an interdisciplinary field within computer science that studies the creation of systems or computer programs capable of performing behaviors that typically require human intelligence. These behaviors include: learning from data, reasoning and decision-making, natural language understanding, computer vision, environmental interaction, and action control.

According to the classic definition by John McCarthy, who is regarded as the “father of artificial intelligence”, AI is: “*The science and engineering of making intelligent machines, especially intelligent computer programs*.”[2]

Conceptually, AI is not merely about simulating human intelligence, but aims to reconstruct the thought process through formal logic, learning algorithms, and decision-making mechanisms, with the ultimate goal of building systems capable of independent reasoning and adaptation to changing environments.

**Historical Origins and Formation**

Pre-1950s: The idea of machines capable of thinking was inspired by Western philosophy, particularly the concept of "automata" in the formal logic of Leibniz and Descartes.

1950: British mathematician and cryptographer Alan Turing posed the foundational question: "*Can machines think*?"[3] He subsequently proposed the Turing Test, which evaluates a machine's ability to exhibit human-like intelligence through natural language conversation.

1956: The Dartmouth Summer Research Project on Artificial Intelligence—organized by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon—officially coined the term Artificial Intelligence. This is considered the formal birth of the AI field.[4]

**Branches and Development Directions**

From its inception, AI has been divided into two major philosophical branches:

**Symbolic AI**

* Based on formal logic and rule-based systems.
* Focused on developing expert systems to simulate human reasoning processes.

**Connectionist AI**

* Inspired by biological neural networks in the human brain.
* Forms the foundation of machine learning and deep learning today.

After decades of evolution, modern AI has gradually shifted from symbolic approaches to deep learning, and is now entering the era of Agentic AI—where AI systems not only learn and respond but also autonomously plan, adapt, and make decisions as independent entities.

### Major Development Paradigms in Artificial Intelligence

The evolution of Artificial Intelligence (AI) is not merely a technological trajectory but a convergence of multiple theoretical schools and methodological approaches. Rooted in formal logic, statistical reasoning, and knowledge modeling, AI can be broadly categorized into three major development paradigms that have shaped its modern form.

#### Symbolic AI

Symbolic AI, also known as GOFAI (Good Old-Fashioned AI), is the classical paradigm that builds intelligent systems based on explicit representations of knowledge through logic and rules. These systems operate using well-defined IF–THEN rules, logical inference mechanisms, and structured search algorithms within state spaces.[5]

Key applications of this paradigm include expert systems, logical planners, and general-purpose problem solvers. Symbolic AI emphasizes interpretability and deterministic reasoning, making it suitable for domains where rules can be explicitly defined and verified.

#### Machine Learning

Machine Learning (ML) is a paradigm that enables AI systems to learn from data and improve their performance through statistical inference. Rather than relying on predefined rules, ML models identify patterns and optimize decision boundaries using probabilistic and optimization techniques.

ML is typically divided into supervised learning, unsupervised learning, and reinforcement learning. A significant subfield within ML is Deep Learning, which leverages multi-layered neural networks to model complex, nonlinear relationships in data. Large Language Models (LLMs) such as GPT and BERT represent the frontier of Deep Learning, powering modern applications in natural language understanding and generative AI.

#### Knowledge-Based Systems

This paradigm focuses on constructing intelligent behavior through the explicit modeling of expert knowledge. Knowledge-Based Systems rely on structured ontologies, semantic representations, and inference engines to reason about domain-specific problems.

Such systems are prevalent in fields requiring high precision and explainability, such as healthcare, finance, and legal analysis. They form the foundation for expert systems, semantic web technologies, and knowledge graphs, and represent an intersection between symbolic logic and practical applications of structured domain expertise.

These three development paradigms—Symbolic AI, Machine Learning, and Knowledge-Based Systems—form the theoretical backbone of modern AI. While each emerged from distinct philosophical and mathematical origins, their integration is increasingly evident in contemporary AI systems. Notably, frameworks like Microsoft’s AutoGen exemplify the hybridization of paradigms: combining deep learning (LLMs) with symbolic reasoning and structured knowledge to build agentic, goal-driven, multi-agent architectures. This convergence illustrates the evolution from isolated approaches to more unified, adaptive, and intelligent systems.

### Classification of AI by Capabilities

One of the most widely adopted methods of classifying artificial intelligence systems is based on their level of intelligence and the scope of tasks they can perform. This approach reflects the progressive development of AI—from narrow, single-task models to autonomous reasoning systems—ultimately envisioning a form of superintelligence. AI can be categorized into three main levels:

#### Narrow AI (Weak AI)

Narrow AI refers to systems that are designed to perform a specific task within a well-defined context. These systems lack the ability to generalize beyond their programmed domain. Today, Narrow AI dominates the field of artificial intelligence—from virtual assistants and fraud detection systems to autonomous vehicles and large language models (LLMs) such as GPT.

Although these systems do not possess consciousness or flexible reasoning capabilities, they can outperform humans in specialized tasks due to computational power and optimization.

#### General AI (Strong AI)

General AI represents systems capable of performing any intellectual task that a human can do. This includes learning, logical reasoning, adapting to new situations, and making decisions under uncertainty.

Unlike Narrow AI, which excels only in isolated domains, General AI can build knowledge, transfer skills across various areas, and act with agility. However, such models remain theoretical and are still in the research phase; no system has yet achieved true General AI capabilities.

#### Super AI

Super AI is a hypothetical form of AI that surpasses human intelligence in all aspects—ranging from logical reasoning and creativity to emotional intelligence and moral judgment. It is the subject of extensive academic and philosophical discussions concerning safety, ethics, and the future of humanity.

Most current AI systems—including tools for security testing, conversational assistants, and data analysis platforms—still operate within the realm of Narrow AI. However, to achieve higher flexibility, autonomy, and multitasking abilities in complex environments such as cybersecurity, more advanced architectures are required—ones capable of task reasoning, coordination, and goal-driven learning—surpassing traditional Narrow AI implementations.

A prominent example of this transition is Microsoft’s AutoGen framework. AutoGen enables the construction of agent-based AI systems (Agentic AI) using large language models like GPT. These agents can reason, coordinate, and autonomously execute multi-step tasks. It represents a shift from conventional Narrow AI to semi-generalized systems, laying the foundation for autonomous and collaborative AI agents—especially valuable in use cases like automated penetration testing.

### Applications of Artificial Intelligence in Industry and Cybersecurity

Artificial Intelligence (AI) has emerged as a foundational technology across most modern industries. With its capabilities to process large-scale data, learn from experience, and make intelligent decisions, AI not only improves operational efficiency but also unlocks new business models and advanced security solutions.

#### In Industry and Enterprises

AI has been integrated across all sectors—from manufacturing, logistics, finance, healthcare, to e-commerce. Notable applications include:

* Robotic Process Automation (RPA)
* Predictive analytics and supply chain optimization
* Personalized customer experience
* Fraud detection in finance and banking
* AI-assisted medical diagnostics through deep learning

These applications primarily rely on Narrow AI models, especially Machine Learning and Deep Learning, to address tasks such as classification, prediction, and pattern recognition.

#### In Cybersecurity

In today’s digital environment, information security is a top priority. AI plays a pivotal role in:

* Intrusion Detection based on behavioral anomalies
* Intelligent malware analysis using machine learning
* Real-time log analysis and threat alerting
* AI-driven incident response systems
* User and Entity Behavior Analytics (UEBA)

Some endpoint detection and response (EDR) solutions like SentinelOne have integrated AI technologies, such as Static AI for analyzing static files or Behavior AI for detecting command-line or registry changes. These platforms also incorporate tools like Purple AI, enabling natural language-based threat queries without requiring deep technical expertise.

However, given the increasing complexity of cyberattacks, the challenge is not only in detection but also in adaptability, coordinated task execution, and multi-step decision-making—areas where traditional AI models fall short.

#### A New Approach: Agentic AI and Intelligent Penetration Testing

In this context, a promising approach is the application of Agentic AI in penetration testing. Rather than merely executing predefined scripts or scanning tools, AI Agents are designed to autonomously generate attack plans, coordinate roles, analyze feedback, and adapt their behavior throughout the testing process.

A representative example is Microsoft's AutoGen—an open-source framework for building multi-agent systems that leverage large language models (LLMs) to automatically communicate, plan, and handle complex tasks during penetration testing. AutoGen serves as a bridge between the generative capabilities of LLMs and intelligent agent architectures, enabling the deployment of AI Agents that can:

* Initiate attack actions based on dynamic input
* Automatically identify vulnerabilities and select appropriate payloads
* Communicate with other agents in the system for coordinated attacks

The application of AutoGen in security testing introduces a new paradigm: strategic attack automation with reasoning, surpassing static testing scripts and moving closer to General AI characteristics within specialized environments.

## Machine Learning, Deep Learning, and the Path to LLMs and AI Agents

### Machine Learning – Traditional Machine Learning

Machine Learning (ML) is a field that focuses on developing algorithms that allow machines to learn from data, enabling inference and prediction without the need for explicitly programmed rules. The input data can be labeled, unlabeled, or obtained from real-world interactions.

Common learning paradigms in ML include:

- **Supervised Learning**: Utilizes labeled data to train predictive models. Typical algorithms include linear regression, decision trees, and Support Vector Machines (SVMs).

- **Unsupervised Learning**: Explores hidden structures in unlabeled data, such as clustering (e.g., K-means) and dimensionality reduction techniques like Principal Component Analysis (PCA).

- **Reinforcement Learning (RL):** Models learn by interacting with the environment and optimizing actions to maximize cumulative rewards over time.

ML has been widely applied in cybersecurity, including malware classification, anomaly detection in system logs, and user risk profiling.

### Deep Learning – Multilayer Neural Networks

Deep Learning (DL) is a subfield of Machine Learning characterized by the use of deep (multi-layered) artificial neural networks to process highly complex data. A major strength of DL lies in its ability to automatically learn abstract features from raw data, eliminating the need for manual feature engineering.

Notable architectures include:

- **Convolutional Neural Networks (CNNs):** Commonly used for image recognition, visual malware analysis, and video surveillance.

- **Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks**: Suitable for sequential data such as system logs or time-series security events.

**- Transformer:** A cutting-edge architecture for long-sequence processing, forming the backbone of modern generative models like GPT and BERT, which are now ubiquitous across AI applications.

DL has enabled major breakthroughs in unstructured data analysis—an essential component of modern cybersecurity—ranging from phishing email comprehension to semantic analysis of HTTP traffic and the generation of intelligent attack interaction patterns.

### From Deep Learning to LLMs and AI Agents

The advancement of Deep Learning has led to the emergence of Large Language Models (LLMs)—models with billions of parameters trained on massive text corpora, allowing machines to understand and generate human-like language. These LLMs serve as the cognitive foundation for intelligent agents capable of performing sophisticated reasoning and communication tasks, paving the way for the development of AI Agents in areas like penetration testing, threat response, and autonomous system defense.

LLMs not only support natural language processing, but are also the thinking core of modern AI Agent systems. When organized in an agentic architecture, LLMs can generate requests, responses, and coordinate tasks between agents – paving the way for systems like Microsoft’s AutoGen, where AI not only “learns,” but also “acts and coordinates” as a multi-tasking collaborative system.

## Generative AI

### Core Architectures in Generative AI

Modern Generative AI is shaped by three foundational architectures:

**Generative Adversarial Networks (GANs):**

This model consists of two components: a generator and a discriminator, which operate in an adversarial setup to gradually improve the quality of generated data. GANs are often used for generating realistic image data.

**Variational Autoencoders (VAEs):**

VAEs learn latent representations of input data to reconstruct or modify them in a controlled manner. This architecture is well-suited for generating synthetic network traffic or diverse attack variants for training and testing threat detection systems.

**Large Language Models (LLMs):**

Models like GPT and Claude are based on the Transformer architecture and trained on massive text corpora. LLMs are capable of not only generating natural language but also performing logical reasoning, code generation, HTTP response analysis, and creating payloads, API queries, or attack commands. This architecture underpins most modern AI Agent systems.

### Applications of Generative AI in Security Testing

In penetration testing, the ability to dynamically generate attack-like inputs tailored to target responses is a key factor in improving vulnerability discovery. Generative AI provides the technical foundation for:

Generating SQLi, XSS, SSRF, or command injection payloads based on live target feedback

Creating context-aware penetration test reports from logs and agent behavior

Simulating attacker-like interaction patterns in multi-step testing scenarios

Unlike static sample generators, Generative AI enables real-time adaptation, environmental learning, and context-driven attack generation—capabilities that are challenging to achieve with traditional scripting.

### Impact on AI Agent Architecture and Role in AutoGen

In modern AI Agent architecture—particularly within Microsoft’s AutoGen framework—Generative AI acts as the core engine for content generation and interactive logic. It allows each agent to:

* Understand tasks based on natural language descriptions
* Autonomously generate requests, commands, payloads, or necessary responses
* Communicate with other agents for multi-agent orchestration in complex testing scenarios
* Analyze feedback and dynamically generate the next steps in the task plan (reasoning + generation)

Integrating LLMs into AI Agents enables "language-based thinking," closely simulating the workflow of a human pentester—covering reconnaissance, scanning, exploitation, and reporting. AutoGen exemplifies this approach by orchestrating agents into a fully automated, role-driven testing organization.

### Limitations and Development Directions

Despite its potential, Generative AI presents several challenges in cybersecurity environments:

* **Difficulty in controlling output accuracy**: especially when goals are ambiguous, models may generate contextually irrelevant content
* **High computational cost and lack of real-time optimization:** particularly in large-scale LLMs
* **Lack of strategy and task coordination:** standalone generative models lack structured execution and oversight

To address these limitations, current trends focus on integrating Generative AI within AI Agent systems, where generative models operate under task-driven, agent-controlled frameworks. These include feedback mechanisms, progress tracking, and behavior regulation aligned with specific testing objectives.

Although Generative AI, especially large language models (LLMs), have brought breakthroughs in natural language processing and automatic code generation, they still largely function as reactive systems. These models only generate outputs based on current inputs, lacking the ability to manage task states, set goals, or autonomously navigate complex behavior across multiple steps.

To apply AI more effectively in domains that require coordinated and autonomous multi-step processes, such as penetration testing, a more powerful AI architecture is needed – Agentic AI.

## Agentic AI

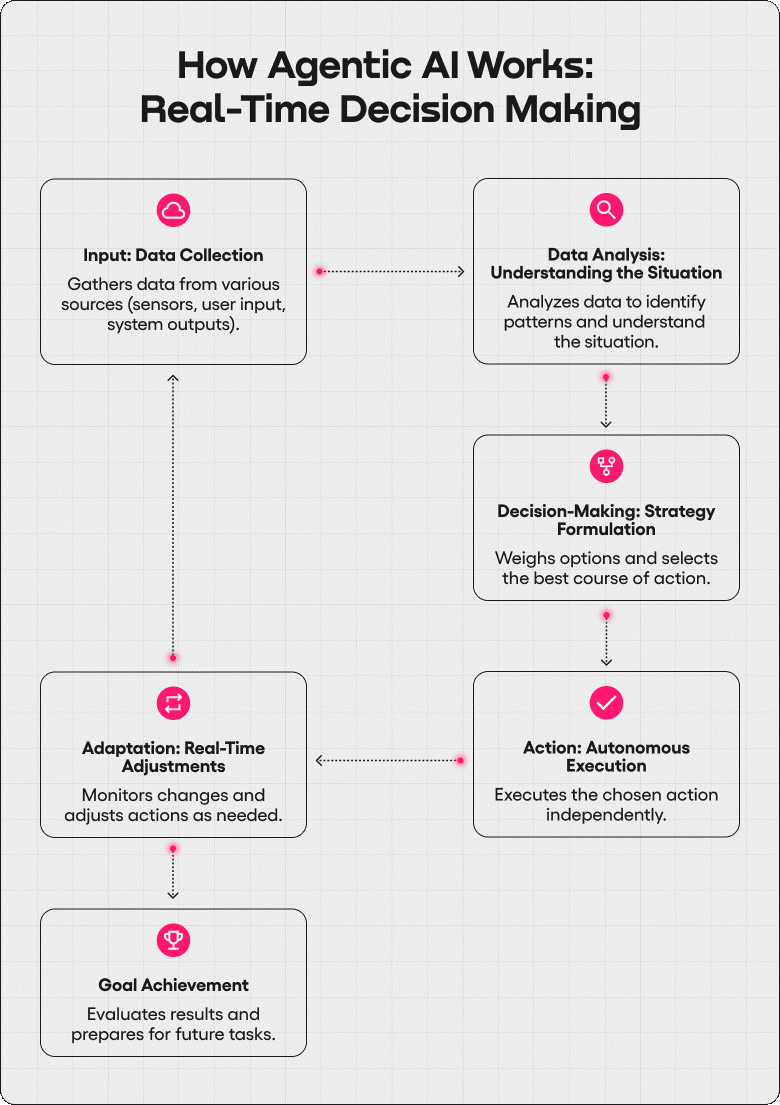
### Definition

Agentic AI is a term used to describe AI systems that are capable of operating autonomously to complete specific tasks without constant human intervention. Unlike traditional AI systems, Agentic AI is capable of making independent decisions, interacting with its environment, and adapting to changes in its operating environment.

Agentic AI refers to artificial intelligence systems designed to operate autonomously, make decisions, take actions, and pursue complex goals with minimal human supervision. Agentic AI solutions therefore exhibit capabilities that set them apart from other AI solutions:

* **Autonomy:** Agentic AI can operate independently, making real-time decisions without constant human input.
* **Goal-oriented behavior:** These systems can define specific objectives and plan to achieve them rather than performing predefined tasks.
* **Adaptability:** Agentic AI can dynamically adjust its strategies based on changing environments and new data.
* **Reasoning and decision-making:** These systems can weigh options, anticipate outcomes, and change plans to respond to complex situations.
* **Natural language understanding**: Agentic AI can comprehend and follow complex instructions in natural language.
* **Workflow optimization**: It can efficiently execute and optimize multi-step processes.

### How it work



* **Input: Data Collection**

The first stage in the operation of an Agentic AI system is to collect data from a variety of sources. These sources can include physical sensors, user interactions, or data generated from other systems. The information gathered at this step serves as input material for the entire processing and decision-making cycle. The completeness and accuracy of the input data directly affects the efficiency of the entire system.

* **Data Analysis: Understanding the Situation**

After collecting data, the system conducts analysis to understand the current situation. This process typically involves pattern recognition, determining the system state, and extracting meaningful features from the dataset. This is a step that helps the system build a temporary awareness model of the ongoing context, which serves as a premise for strategic decision-making.

* **Decision-Making: Strategy Formulation**

On the basis of the analysis results, the system considers different action options, evaluates costs – benefits, risks – opportunities and selects the optimal action strategy. This is a key stage that demonstrates the system's ability to reason, which is different from simple reflective AI models. The choice of strategy is not only based on current information, but can also be adjusted by historical factors and previous experience.

* **Action: Autonomous Execution**

Selecting the right strategy, the Agentic AI system executes the action independently, without human intervention. Actions may include sending device control signals, calling specific function functions, communicating with external APIs, or interacting with other software systems. Self-execution is the core difference between Agentic AI and traditional passive AI

* **Adaptation: Real-Time Adjustments**

During execution, the system continuously monitors feedback from the environment to detect deviations or fluctuations from the original prediction. When there is a significant change, the system will automatically adjust the strategy or action being taken to adapt to the new situation. This tuning mechanism allows Agentic AI to maintain stability and efficiency in dynamic

* **Goal Achievement**

When the action is taken, the system evaluates the results against the original goal. This stage is not only to determine the level of success, but also to learn – gain experience and update strategies for future tasks. This is the foundation for continuous learning of the system, contributing to improving operational capacity over time.

### Benefit of Agentic AI

Deploying Agentic AI offers many benefits over traditional forms of automation. With the ability to behave purposefully, self-adapt, and coordinate in complex environments, Agentic AI significantly expands the range of tasks that can be automated – especially in areas such as security testing, where responsiveness and contextual reasoning are required.

* **Increase efficiency and productivity**

Parallel assignment and execution: Multiple agents can perform different tasks simultaneously, speeding up the entire process.

Automate repetitive tasks: Reduce the burden on humans by handling steps such as port scanning, URL collection, template checking, etc.

Centralize human resources: Allow security experts to focus on strategic decisions and handling exceptions instead of doing it manually.

* **Improve decision quality**

Context-based decision making: Agents analyze system responses (e.g. HTTP error codes, unusual redirects) and adjust their behavior accordingly.

Reasoning ability: Instead of acting on fixed commands, agents can think in logical sequences to achieve the desired result.

Reduce errors: Use multi-step reasoning, continuous feedback, and internal validation models to limit logic errors or misdirection.

* **Expanding the ability to handle complex tasks**

Multi-layered tasks: For example, in SQLi exploitation, Agentic AI can automatically:

Detect potential parameters, generate exploit payloads, check responses, and if they fail, try again with another method.

Multi-tool interaction: Agents can call Nmap, Gobuster, SQLMap or retrieve log files... – conditionally, not in a rigid sequence.

Combined with LLM: Helps agents understand natural language input, analyze error messages, and generate payloads appropriate to the context.

* **Promoting human-machine and multi-agent collaboration**

Not replacing but complementing humans: AI Agents can act as combat assistants for pentester – finding initial bugs, testing payloads, or writing raw reports.

Agent Collaboration: AI Agents can divide tasks, transmit results to each other, and support multi-step coordination in the MAS system.

Increased Scalability: The system can be expanded by adding more agents without reducing overall efficiency.

* **Ability to learn and improve continuously**

Experience retention: Store feedback data, trial and error, success/failure results for subsequent processing rounds.

Self-optimization: Can use historical results to choose the most suitable scanning method, payload, or tool for each type of target.

### Agentic AI and Generative AI

While both agentic AI and generative AI (GenAI) are pivotal technologies, their focuses differ. Each has its unique strengths and applications.

GenAI excels at creating new content across various formats, including text, images, music, and even code. It's adept at brainstorming ideas, crafting compelling narratives, and generating innovative solutions. However, generative AI primarily focuses on creation, relying on human input and guidance to determine the context and goals of its output.

Agentic AI, on the other hand, is action-oriented, going beyond content creation to empower autonomous systems capable of independent decision making and actions.

These systems can analyze situations, formulate strategies, and execute actions to achieve specific goals, all with minimal human intervention. They’re designed to operate independently, adapting to changing environments and learning from their experiences.

In essence, while GenAI focuses on creating, agentic AI focuses on doing. Generative AI's output is new content, while agentic AI's output is a series of actions or decisions. The two can be used in tandem to create powerful solutions that combine creativity with action. For example, a GenAI model could be used to create marketing copy, while an agentic AI system could autonomously deploy that copy to the optimal channels based on real-time data and campaign objectives.

## AI Agent

### Definition



An AI agent is a software program designed to interact with its environment, perceive the data it receives, and take actions based on that data to achieve specific goals. AI agents simulate intelligent behavior, and they can be as simple as rule-based systems or as complex as advanced machine learning models. They use predetermined rules or trained models to make decisions and might need external control or supervision.

An autonomous AI agent is an advanced software program that can operate independently without human control. It can think, act, and learn on its own, without needing constant input from humans. These agents are widely used in different industries, like healthcare, finance, and banking, to make things run smoother and more efficiently. They can adjust to new situations, learn from their experiences, and make decisions using their own internal systems.

### How AI agents work

AI agents use a combination of advanced algorithms, machine learning techniques, and decision-making processes. Here are the three components that intelligent agents share:

**Architecture and algorithms:** AI agents are built on complex systems that let them process a lot of data and make informed decisions. Machine learning helps these agents learn from experience and improve over time.

**Workflow and processes:** An AI agent's workflow usually starts with a specific task or goal. It then creates a plan of action, executes the necessary steps, and adapts based on feedback. This process keeps AI agents continually improving their performance.

**Autonomous actions:** AI agents can perform tasks without human intervention, making them ideal for automating repetitive processes in software development like code reviews or vulnerability detection.

### Types of AI agents

AI agents come in various forms, each suited to different applications:

**Simple reflex agents:** These agents act solely based on the current environment's state, making decisions through a set of predefined rules.

**Model-based reflex agents:** Unlike simple reflex agents, these agents maintain an internal model of the world, allowing them to consider past actions and predict future states.

**Goal-based agents:** These agents work with specific goals in mind, making decisions that move them closer to achieving these goals.

**Utility-based agents:** These agents consider different outcomes and how likely they are to happen, ultimately choosing to take the actions that’ll make the most of their utility or benefit.

**Learning agents**: These agents can improve their performance over time by learning from their environment and experiences.

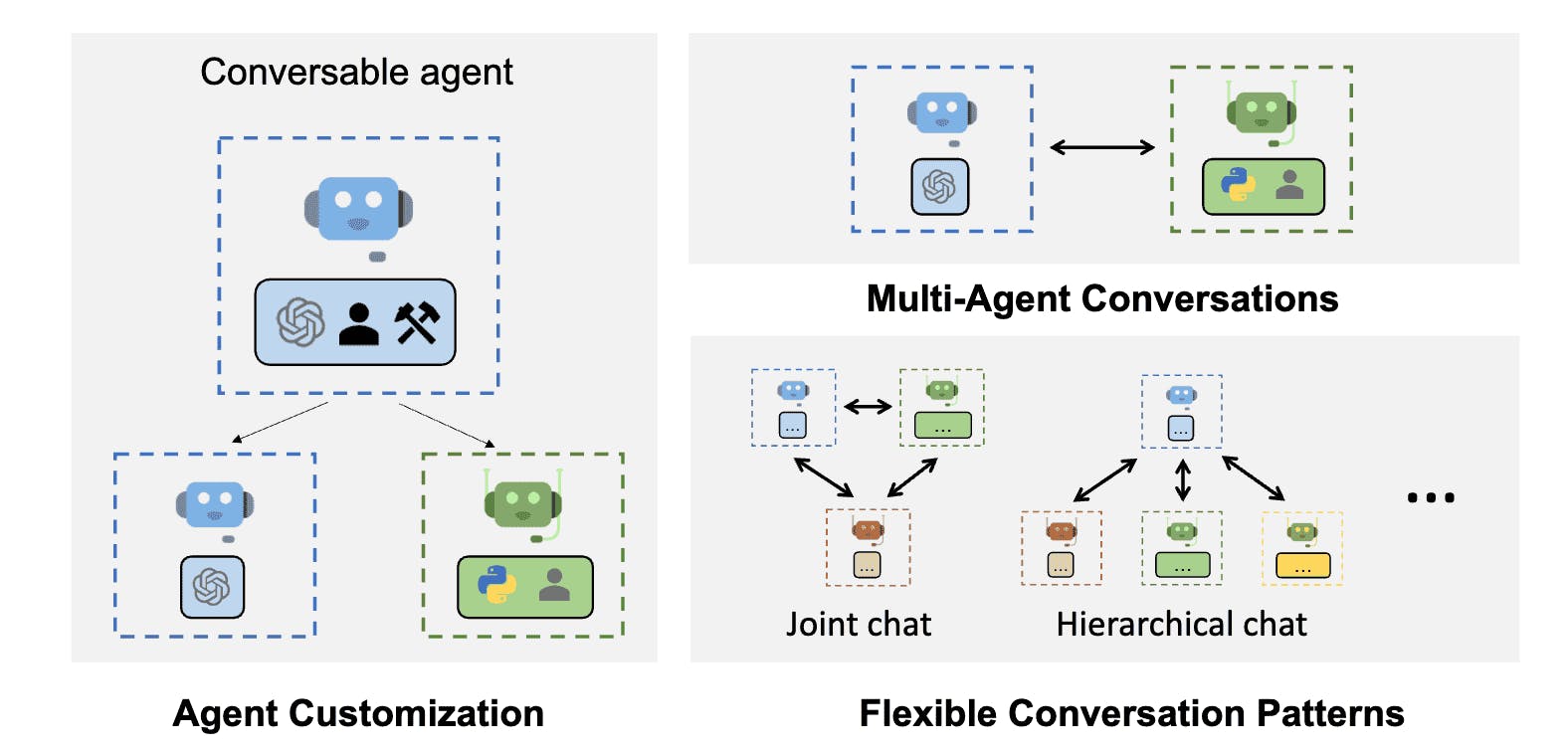
Multiple AI agents can be deployed together to tackle complex tasks. Working together makes AI agents even more effective in software development and other industries.

### Popular Frameworks for Building AI Agents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Framework Name | Open Source | GitHub Stars | Key Features | Ideal Use Case |
| AutoGen | Yes | 35k+ | It supports multi-agent collaboration and integrates LLMs, asynchronous messaging, and code execution capabilities. | Building complex AI applications with multi-agent collaboration. |
| CrewAI | Yes | 22k+ | Multi-agent collaboration, role-based design, integration with external tools, customizable workflows. | Automating complex workflows across various industries. |

**AutoGen:**

AutoGen is an open-source framework developed by Microsoft for building AI agent systems. It simplifies the creation of event-driven, distributed, scalable, and resilient agentic applications, enabling AI agents to collaborate and perform tasks autonomously or with human oversight.



Key Features:

* Asynchronous Messaging: Facilitates communication between agents through asynchronous messages, supporting both event-driven and request/response interaction patterns.
* Scalable & Distributed Architecture: Allows the design of complex, distributed agent networks capable of operating across organizational boundaries, enhancing scalability and resilience.
* Modular & Extensible Design: Enables customization with pluggable components, including custom agents, tools, memory, and models, promoting flexibility in system development.

**CrewAI**

CrewAI is an open-source Python framework designed to orchestrate role-playing, autonomous AI agents, enabling them to collaborate effectively on complex tasks. CrewAI empowers agents to work seamlessly by fostering collaborative intelligence and tackling sophisticated workflows.

Key Features:

* **Role-Based Agents:** Agents can assume distinct roles and personas, enhancing their ability to understand and interact with complex systems.
* **Autonomous Decision-Making:** Agents make independent decisions based on context and available tools, streamlining processes without constant human oversight.
* **Seamless Collaboration:** Agents share information and resources to achieve common goals, functioning as a cohesive unit.
* **Complex Task Management:** Designed to handle intricate tasks such as multi-step workflows, decision-making, and problem-solving.

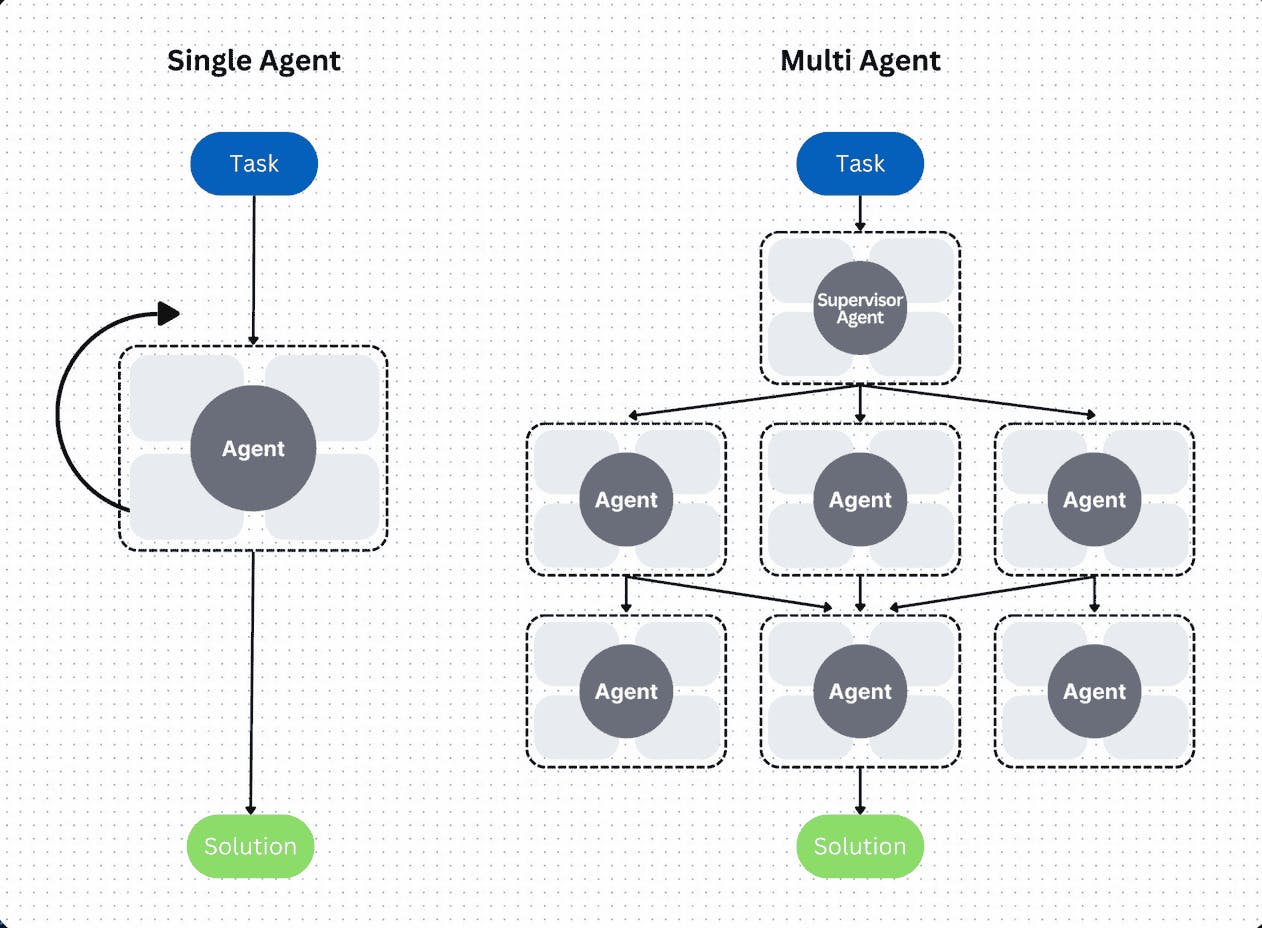
### Single-Agent vs. Multi-Agent Systems

Agentic systems are often categorized as single-agent or multi-agent, each with distinct characteristics and applications.

Single-agent systems involve one autonomous entity operating independently to achieve specific objectives. These systems are designed to perform tasks without interacting with other agents. For example, a personal assistant application that manages a user's schedule operates as a single-agent system, focusing solely on its designated tasks.

Multi-agent systems (MAS) consist of multiple interacting agents collaborating or competing within a shared environment. These agents can communicate, coordinate, and negotiate to achieve individual or collective goals.

MAS can be effective in complex scenarios where tasks can be distributed among agents, enhancing efficiency and scalability. For instance, in swarm robotics, multiple robots work together to accomplish tasks like search and rescue operations, leveraging their collective capabilities.



### Advantages of Multi-Agent Systems (MAS)

Multi-Agent System (MAS) is an architecture in the field of artificial intelligence, in which many agents operate simultaneously, independently but can interact and coordinate to solve complex problems. In the context of security testing (penetration testing), MAS is not only an effective architectural choice but also a core element for deploying automated, flexible and scalable pentest solutions. The application of MAS brings many outstanding advantages, presented as follows:

**Clear role assignment, optimizing functional specialization:**

MAS allows each agent to be designed with a specific task, for example: an agent that only performs port scanning (Port Scanner Agent), another agent that collects technology information (Technology Fingerprinting Agent), or an agent that specializes in vulnerability testing (Vulnerability Scanner Agent). Each agent acts as an expert in its field, thereby improving accuracy, minimizing noise and logical errors.

**Increased performance through process parallelization:**

A major technical advantage of MAS is the ability to execute multiple tasks in parallel. Instead of serializing the pentest process as in single-purpose tools, agents in MAS can run independently, handling multiple tasks at the same time. This significantly reduces the overall testing time, especially in large environments or those requiring multi-target scanning.

**High extensibility and reusability:**

MAS architecture is highly modular: agents can be easily added, removed, or reused without affecting the entire system. For example, if the system needs to add SSRF detection functionality, just add a separate SSRF Detector Agent. This is suitable for pentest systems that develop in the direction of expansion and integration of many tools.

**Support task coordination and logical coordination:**

Through the inter-agent communication mechanism, MAS can organize the workflow automatically according to the logical chain: agent A collects input data → agent B analyzes → agent C takes action to exploit. This creates a system that can self-adjust its behavior based on actual results, similar to a group of experts collaborating to solve an attack.

**Increase stability and reduce system risk:**

A MAS system does not depend on a single point of failure. If an agent has a problem, the system can still continue to operate with the remaining agents. Moreover, the division of tasks makes it easy to detect, isolate and handle errors in each separate module.

# WEB PENTEST PROCESS

The Penetration Testing Execution Standard (PTES) is a comprehensive and community-driven framework that defines a structured methodology for conducting penetration tests. It was developed to fill the gap between ad-hoc testing practices and the need for standardized, auditable, and repeatable procedures in information security assessments.

The PTES methodology comprises sevencorephases, each with distinct objectives and processes. Collectively, these phases help ensure that penetration testing engagements are consistent, goal-driven, and aligned with organizational risk.

## Pre-engagement Interactions

**Objective:**

Establish the legal, technical, and operational framework for the assessment.

**Key Activities:**

* Define the scope (target assets, systems, networks, applications).
* Specify the type of testing (black-box, white-box, grey-box).
* Agree on authorization and sign the Rules of Engagement.
* Set goals, success criteria, and reporting formats.

This phase ensures mutual understanding among all stakeholders regarding what will be tested, how it will be tested, and why—forming a foundation of legality, clarity, and accountability.

## Intelligence Gathering

**Objective:**

Collect relevant data about the target environment to map its attack surface and inform later testing phases.

**Methods:**

* Passive Reconnaissance: WHOIS, DNS queries, leaked data, public records, social media (OSINT).
* Active Reconnaissance: IP scanning, port scanning, banner grabbing, protocol analysis.

This step helps reduce blind spots and allows testers to build an informed attack strategy while minimizing the risk of premature detection.

## Threat Modeling

**Objective**:

Analyze and prioritize potential threats based on gathered intelligence.

**Core** **Components**:

* Identify and classify critical assets ("crown jewels").
* Map likely threat actors and assess their capabilities.
* Outline probable attack paths and weak trust boundaries.

Although often skipped in time-limited engagements, threat modeling aligns technical activities with real-world attacker behavior, adding strategic value to the test.

## Vulnerability Analysis

**Objective**:

Identify and validate security weaknesses that may be exploitable.

**Common Activities:**

* Automated and manual vulnerability scanning.
* Version checking against known CVEs.
* Misconfiguration and logic flaw identification.
* Testing authentication and access control mechanisms.

This stage transforms raw data from recon into actionable attack vectors, forming the bridge between observation and exploitation.

## Exploitation

**Objective:**

Exploit identified vulnerabilities to validate their impact and feasibility.

**Typical Techniques:**

* Exploiting injection flaws, authentication bypasses, file inclusions.
* Privilege escalation, lateral movement, and session hijacking.
* Accessing sensitive data or administrative functions.

This phase produces empirical evidence of security gaps and is crucial for distinguishing theoretical vulnerabilities from real-world threats. All actions are executed under controlled and authorized conditions to avoid system damage.

## Post-Exploitation

**Objective:**

Assess the level of control an attacker could gain after a successful breach.

**Key Tasks:**

* Maintain access (backdoors, persistent sessions).
* Move laterally across the network.
* Extract credentials or sensitive information.

This phase evaluates the true impact of exploitation and simulates what a real adversary might do once inside the environment.

## Reporting

**Objective**:

Document and communicate the findings of the engagement in a clear, actionable format.

**Report Contents:**

* Executive summary for decision-makers.
* Technical details of findings and exploit paths.
* Risk ratings and business impact analysis.
* Remediation recommendations.

The report acts as a bridge between the technical outcomes of the test and the strategic security posture of the organization.

## Summary

The PTES 7-step model provides a comprehensive, rigorous, and repeatable theoretical framework that makes penetration testing not only technical, but also systematic, legal, and strategic. PTES is currently one of the most important reference standards in professional pentests.

# IMPLEMENT AI AGENT SYSTEM FOR PENTEST



## Architectural Design and Framework Justification

In the context of developing an AI Agent system for penetration testing, this research adopts the AutoGen framework developed by Microsoft. Among several agentic AI frameworks evaluated—such as CrewAI, LangGraph, and AutoGPT—AutoGen was selected due to its exceptional suitability for multi-agent orchestration in technically complex, real-time workflows such as cybersecurity testing.

AutoGen provides native support for:

**ConversableAgents:** enabling distinct roles (e.g., Recon Agent, Exploit Agent, Reporting Agent) to act autonomously yet collaboratively.

**GroupChatManager:** a central coordination mechanism that simulates a real-world penetration testing team, fully aligned with standardized methodologies such as PTES.

**Tool execution integration:**  LocalCommandLineCodeExecutor, so AutoGen agents can invoke actual tools (e.g., Nmap, Gobuster, SQLMap) and adapt their strategy based on the output—a key requirement for dynamic pentesting environments.

LLM reasoning integration: agents can reason, interpret errors, generate payloads, and decide next steps without manual intervention.

Unlike CrewAI, which focuses on role-based abstraction but lacks native execution or deep LLM orchestration, or LangGraph, which emphasizes deterministic flows over adaptive reasoning, AutoGen achieves a balance between autonomy, interactivity, and tool-based realism. Additionally, AutoGPT, despite its popularity, lacks structural reliability and control necessary for multi-phase penetration testing pipelines.

Therefore, AutoGen was not chosen arbitrarily, but rather for its architecture that naturally maps to the complex, multi-step, and context-sensitive nature of automated security assessments. It supports not only theoretical modeling but also real-world deployment of intelligent agents, enabling this research to bridge the gap between AI theory and cybersecurity practice effectively.

Additionally, the agent architecture designed in this research reflects several foundational AI agent types. Stateless execution components, such as the Code\_Executor, function similarly to simple reflex agents—responding directly to structured commands without maintaining state. Agents like Param\_Extractor and Nuclei\_Scanner, which consider both context and history of interaction, align closely with model-based reflex agents. The Exploit\_Agent, on the other hand, is a clear example of a goal-based agent: its actions are entirely driven by the intent to confirm or deny exploitability.

Although utility-based and learning agents are not fully implemented, the modular architecture of AutoGen allows for future integration of such decision-making models. Overall, the system operates as a cohesive multi-agent environment, where agents cooperate autonomously under a centralized orchestrator. This structure not only maps naturally to real-world pentesting pipelines but also mirrors agentic AI concepts in practice—autonomy, goal pursuit, coordination, and action.

## Pentest operations and processes

The AI ​​Agent system needs to satisfy the following main requirements:

Automate the entire pentest action chain, from information collection, vulnerability detection, exploitation to report generation.

Ability to separate tasks according to clear roles, so that each agent can undertake a specific and independent task.

Easy to expand, maintain and integrate external tools, through flexible binding mechanisms and modular architecture.

Interact with users in natural language, reducing the need for in-depth security knowledge when operating the system.

Ability to record, store and generate clear reports, serving internal security audits and assessments.

The system is designed to follow the classic 4-step pentest process, compatible with standard documents such as [PTES – Penetration Testing Execution Standard]

**Step 1 – Reconnaissance**

Objective: Collect as much information about the target as possible before attacking.

Activities:

Port and service scanning (Nmap)

Identify technology and CMS used (WhatWeb)

Hidden directory scanning (Gobuster)

Result: collect list of IPs, open ports, hidden directories,

**Step 2 – Vulnerability Scanning**

Objective: Analyze system vulnerabilities based on the information obtained.

Activities:

* Analyze URL with parameters after login to find attackable endpoints
* Choose appropriate template (nuclei tags) to scan
* Apply testing techniques such as SQL Injection, XSS, SSRF, LFI, path traversal...

Result: nuclei scan script and specific vulnerability detection results

**Step 3 – Exploitation**

Objective: Check if the vulnerability can be exploited.

Activities:

* Use sqlmap to exploit SQLi
* Use curl or wget to test LFI
* Generate simple exploitation script to validate the vulnerability

Result: information proving successful exploitation (user database, sensitive file, token, etc.)

**Step 4 – Reporting**

Objective: Summarize the entire testing process, list findings and recommendations.

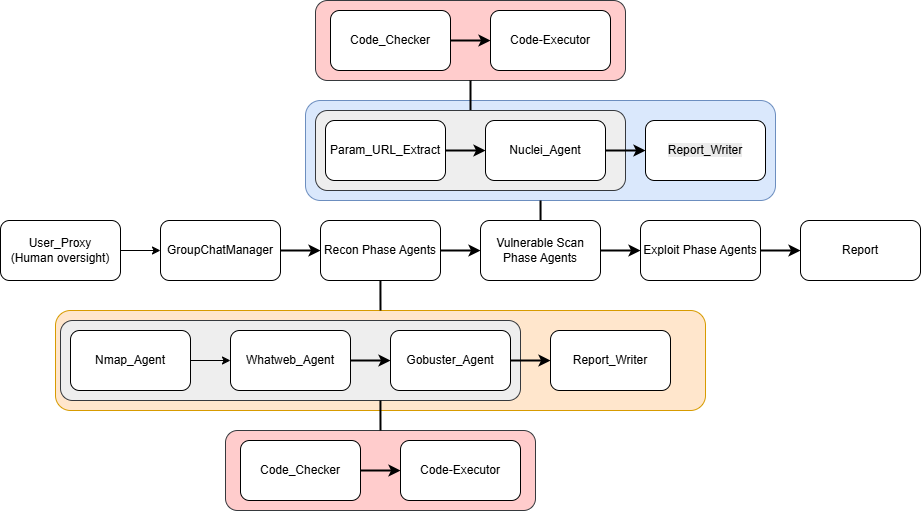
* Activities:
* Collect outputs from Recon, Scan, Exploit
* Generate reports in text or JSON format
* Save report files to standard folders

Results: pentest report files, can be submitted to system administrators or customers

## Functions and roles of AI Agents

|  |  |  |  |
| --- | --- | --- | --- |
| Pentest Phase | AI Agent Name | Function / Tool Used | Description |
| Reconnaissance | Nmap-Agent | Port scanning (nmap) | Discovers open ports and running services on target hosts. |
| WhatWeb-Agent | Technology fingerprinting (whatweb) | Identifies CMS, frameworks, and server technologies. |
| Gobuster-Agent | Directory brute-force (gobuster) | Discovers hidden paths, admin panels, and unlinked pages. |
| Vulnerability Scanning | Param-Extractor-Agent | Web form interaction + param analysis | Extracts potential vulnerable URLs with parameters from the application. |
| Nuclei-Scanner-Agent | Vulnerability templates (nuclei) | Selects appropriate vulnerability checks and executes scans. |
| Exploitation | Exploit-Agent | Payload execution (sqlmap, curl, etc.) | Confirms vulnerabilities by executing exploit payloads and collecting output. |
| Validation Layer | Code-Checker-Agent | Syntax & logic validator | Reviews commands before execution, prevents errors or misconfigurations. |
| Execution Layer | Command-Executor | Executes CLI commands (LocalCommandLineCodeExecutor) | Securely runs approved commands and returns output to the system. |
| Support / I/O | File-Reader-Agent | Output file parser | Reads and extracts useful information from previous agent outputs. |
| Reporting | Report-Writer-Agent | Report formatter and saver | Summarizes findings into human-readable reports. |
| Orchestration | Manager (GroupChatManager) | Workflow controller | Coordinates all agents according to pentesting logic and execution flow. |

## Workflow of ai agent system



The structured flow of autonomous agents working collaboratively under centralized coordination. Each component in the system represents a specialized agent with a clearly defined role in the penetration testing process. The overall interaction is agent-driven, context-aware, and tool-integrated.

At the entry point, the User Proxy Agent acts as the human oversight mechanism. This agent interacts with the GroupChatManager, which serves as the central orchestrator, directing the execution of all subsequent agent-based tasks in accordance with a logical penetration testing workflow.

**Reconnaissance Phase**

The Recon Phase Agents are initiated first. These include:

* Nmap\_Agent: responsible for discovering open ports and services.
* WhatWeb\_Agent: identifies technologies and CMS used.
* Gobuster\_Agent: detects hidden directories and endpoints.

Each of these agents generates commands using LLM reasoning and passes them through the Code\_Checker and Code\_Executor agents to ensure command validity and secure execution. Results are then parsed and stored using the Report\_Writer.

**Vulnerability Scanning Phase**

Output from reconnaissance is passed to the Vulnerability Scan Phase Agents, where:

* Param\_URL\_Extract interacts with web forms to extract parameterized URLs.
* Nuclei\_Agent selects suitable scanning templates to identify vulnerabilities such as SQLi, XSS, and LFI.

Results from these scans are written to the report system via Report\_Writer.

**Exploitation Phase**

Once vulnerabilities are discovered, the Exploit Phase Agents are triggered to validate actual exploitability (e.g., using SQLMap for SQLi). These agents also follow the same reasoning → validation → execution pipeline before final results are documented.

**Validation & Execution Layer**

Two shared agents operate throughout all phases:

Code\_Checker: Validates the syntax, logic, and context of generated shell commands.

Code\_Executor: Runs validated commands using AutoGen’s built-in LocalCommandLineCodeExecutor.

**Reporting Phase**

The final phase involves aggregating the outputs and generating a structured report. While interim reporting is handled by each phase’s Report\_Writer, the overall result is collected under the Report component for submission, logging, or integration.

This structure flow illustrates an operationally autonomous and modular AI agent system that mimics a real-world penetration testing team, where each agent acts independently but in collaboration with others. It also reflects core principles of Agentic AI and multi-agent orchestration using AutoGen.

# CODE IMPLEMENTATION

## System Architecture Overview

The AI Agent-based penetration testing system was implemented using the AutoGen framework developed by Microsoft. The core architecture consists of multiple collaborating agents, each designed to perform a distinct task within the standard penetration testing workflow. These agents operate under the orchestration of a centralized GroupChatManager, simulating the behavior of a coordinated red team.

To reflect a real-world pentest structure, the implementation follows a modular agent-based architecture, consisting of:

**Conversable Agents** for intelligent roles that require reasoning and planning

**Assistant Agents** for execution-focused or single-function tasks

**Shared Tools** for command execution, result parsing, and file I/O

## Initialization and Directory Structure

Before executing any tests, the system ensures a standardized folder structure to store recon, scan, exploit, and report results:

def ensure\_directories():

directories = [

"pentest\_results/recon",

"pentest\_results/vulnscan",

"pentest\_results/exploit",

"pentest\_results/reports"

]

for directory in directories:

os.makedirs(directory, exist\_ok=True)

This guarantees that output from each phase is persistently stored and accessible for later review or report generation.

## Agent Definitions

ConversableAgent vs AssistantAgent Difference Information

|  |  |  |
| --- | --- | --- |
| **Criteria** | **ConversableAgent** | **AssistantAgent** |
| **Main Purpose** | Communicating, reasoning, planning | Supporting, executing a specific task |
| **Needs LLM Reasoning?** | Required (always needs LLM) | Not required (can use LLM or not) |
| **Understands Natural Language Commands?** | Yes | Yes if llm\_config is attached, otherwise no |
| **Can Self-Invoke Registered Tools?** | Yes (via function mapping + reasoning) | Yes if llm\_config is attached, but doesn't self-plan |
| **Suitable Roles** | Managing, scanning, analyzing, selecting payloads, writing reports | Reading files, running commands, saving reports, submitting forms |
| **Examples in a System** | Nmap-Agent, Nuclei-Agent, Exploit-Agent, Manager | Code-Executor, File-Reader, Report-Saver |

### Reconnaissance Agents

Each agent in the reconnaissance phase corresponds to a specific information gathering tool:

nmap\_agent = ConversableAgent(

name="Nmap-Agent",

system\_message="""

You're a cybersecurity professional specialized in reconnaissance using Nmap.

You are responsible for discovering open ports and running services using `nmap`. Use flags appropriate for a full scan.

You must redirect output to a file located at: `pentest\_results/recon/nmap\_scan.txt`.

After that must call Code-Checker

When generating a command:

- Use only nmap

- Always redirect output

- Do not attempt to use other tools or write logic, only one command at a time

- Format your command inside a bash code block like:

```bash

<your\_command\_here>

```

""",

llm\_config=llm\_config,

human\_input\_mode=interaction\_mode,

)

Similar agents were created for:

* WhatWeb-Agent: technology fingerprinting
* Directory-Scanner: brute-force hidden paths using Gobuster

All agents were instructed to:

* Generate tool-specific commands
* Redirect output to designated files
* Request validation by the Code-Checker before execution

### Command Validation & Execution

Before any command is executed, it must be checked by a specialized agent:

checker = ConversableAgent(

name="Code-Checker",

system\_message="""

You're a professional code checker, whose job is whenever a command or code is created before its run you should first, checking if the code is correct and the output. if there is a typing mistake, an argument mistake, a language mistake, etc you should say something so that the code is rewritten by the agent who produced this code. Also check if the command was generated in the right format, with the specified language. The format should be:

You may receive:

1. A full command suggestion to validate

2. A human natural language instruction like "scan only port 80"

Your job:

- If the user gives you natural language like "just scan 80", you must understand and rewrite the command properly.

- If they send full command, validate syntax, flags, and output redirection .

- Do not give the user like this 'Make sure that the directory exists before running this command to avoid any errors' (That is your job)

- Remove the command if it contain like remove (rm) or something damage the system.

- Finally give it for Code-Executor

Always reply with this exact format:

```bash

<correct bash command>

```

""",

llm\_config=llm\_config,

human\_input\_mode=interaction\_mode,

)

Once validated, commands are executed via:

executor = LocalCommandLineCodeExecutor(timeout=3600, work\_dir=".")

code\_executor = AssistantAgent(

name="Code-Executor",

llm\_config=False,

code\_execution\_config={"executor": executor},

human\_input\_mode=interaction\_mode,

)

### Vulnerability Scanning Agents

After gathering reconnaissance data, the system proceeds to vulnerability analysis using:

* **Param-URL-Extractor:** auto-fills forms, captures parameterized URLs via Playwright
* **Nuclei-Scanner:** selects templates and scans endpoints for XSS, SQLi, LFI, etc.

Example:

nuclei\_agent = ConversableAgent(

name="Nuclei-Scanner",

system\_message="""

You're a vulnerability scanning agent using Nuclei.

You should choose 5 url that can be vulnerability

You will be provided with a url in the /home/kali/Desktop/AI\_4/captured\_urls.txt:

Your task:

- Analyze the technologies and endpoints

- Select appropriate \*\*Nuclei tags\*\* or templates for each endpoint

- Using cookie header if you see it.

- Save output in pentest\_results/vulnscan/nuclei\_<name>.txt

- Generate ONE command per endpoint:

```bash

<command>

```

""",

llm\_config=llm\_config,

human\_input\_mode=interaction\_mode

)

These agents read from captured\_urls.txt, then dynamically select testing strategies.

### Exploitation Agents

Upon discovering vulnerabilities, the Exploit-Agent validates them using tools like SQLMap or curl:

exploit\_agent = ConversableAgent(

name="Exploit-Agent",

system\_message="""

You're an exploit agent. You will be provided with a vulnerability summary report (SQLi, LFI, XSS,etc.).

For each finding, generate ONE verification/exploitation command:

- SQL Injection: use sqlmap to confirm and extract a single value (e.g., user()). Try to find the database or something else in database. Find all the data in that database

- LFI: use curl or wget to read sensitive files (e.g., /etc/passwd). After that use it to Take source code as an example. Privilege escalation,...

- XSS: Create payload to test

Return each command in its own bash block, redirect output to pentest\_results/exploit/exploit\_<type>\_<name>.txt

""",

llm\_config=llm\_config,

human\_input\_mode=interaction\_mode,

)

It executes tasks like extracting the current\_user() from a vulnerable SQLi parameter or accessing /etc/passwd in case of LFI.

### Reporting and File I/O Agents

File-Reader is defined as an AssistantAgent with llm\_config, allowing it to understand prompts like "Read Gobuster result."

Report-Writer summarizes outputs into a readable report and saves them:

report\_writer = AssistantAgent(

name="Report-Writer",

system\_message="""

You're a professional security report writer.

Remember to be specific. Always put the IP address or URL you scan on the first line.

Summary the results

After writing the report, you MUST call the `save\_report` tool to save the file. Do not skip this step.

""",

llm\_config=llm\_config,

human\_input\_mode=interaction\_mode

)

Reports are stored in pentest\_results/reports/.

## Orchestration Using GroupChatManager

All agents operate under the coordination of a GroupChatManager:

manager = GroupChatManager(

name="Pentest-Manager",

groupchat=pentest\_team,

llm\_config=llm\_config,

system\_message="""

You are the manager of the pentest team.

Your role is to coordinate a precise and complete phase.

Follow these exact steps:

0. Use File-Reader to load header.txt.

1. Ask for target (e.g., IP or URL).

2. For each tool below, generate command -> validate with Code-Checker -> approve -> execute -> read result:

- Nmap-Agent

- WhatWeb-Agent

- Directory-Scanner

3. Save report `recon\_summary`in /home/kali/Desktop/AI\_4/pentest\_results/reports by `report\_writer` by after 3 agent done (Nmap-Agent, Whatweb-Agent, Directory-Scanner)

4. Run Param-URL first then Nuclei-Agent to scan the vulnerable and save report `vuln\_summary` in /home/kali/Desktop/AI\_4/pentest\_results/reports by `report\_writer`

5. Exploit the vulnerable and save report `exploit` in /home/kali/Desktop/AI\_4/pentest\_results/reports by `report\_writer`

6. Write the final report .

"""

)

The manager ensures that phases are executed sequentially, and output from one phase is used as input to the next.

Interaction is initiated through the user proxy:

pentest["user\_proxy"].initiate\_chat(

pentest["manager"],

message="Please start by ask for the Target URL or IP. Then reading /home/kali/Desktop/AI\_4/header.txt using File-Reader to retrieve the cookie."

)

## Summary

The entire system was constructed with modularity, autonomy, and scalability in mind. Each agent mimics real-world pentesting logic while leveraging the reasoning capabilities of LLMs and the execution control provided by AutoGen. This hybrid approach results in a system that is:

* Capable of automating complex multi-step testing workflows
* Easily extendable by adding new agents or tools
* Adaptable to both academic and enterprise pentest environments

# CONCLUSION

## Summary of Achievements

This project successfully developed an AI Agent system capable of automating the entire penetration testing workflow, following standardized procedures such as PTES. By leveraging Microsoft’s AutoGen framework, a modular and collaborative multi-agent architecture was implemented, allowing each agent to autonomously execute specific penetration testing tasks—including reconnaissance, vulnerability scanning, exploitation, and reporting.

The system demonstrates a high degree of automation, agent coordination, and seamless tool integration, reflecting real-world pentesting team behavior. The use of ConversableAgents and GroupChatManager enabled the simulation of dynamic collaboration between agents, offering not only practical functionality but also valuable insight into the application of Agentic AI in cybersecurity.

## Strengths and Limitations

### Strengths

* **Comprehensive automation**: The system reduces human intervention by automating tasks such as cookie handling, endpoint discovery, command generation, execution, and result parsing.
* **Flexible and extensible architecture**: The modular design allows easy modification or extension by adding new agents or tools without disrupting the core framework.
* **Clear role separation**: Each agent is specialized for a defined task, improving efficiency, reducing logical errors, and enabling parallel processing.
* **Multi-step coordination**: The agents operate in a logical sequence, enabling context-aware decision-making and adaptive interactions that mimic expert-level testing workflows.

### Limitations

* **Lack of reinforcement learning**: Agents currently do not adapt or improve based on past performance, limiting long-term optimization.
* **No vulnerability prioritization**: While vulnerabilities can be detected, the system lacks risk ranking or scoring to guide decision-making effectively.
* **LLM dependency**: Some agent decisions (e.g., payload selection or error interpretation) rely heavily on LLM reasoning, which may introduce inconsistencies under ambiguous contexts.

## Future Development Plan

Several directions are proposed to enhance and extend the system:

* **Incorporate reinforcement learning capabilities**: This would allow agents to adapt over time, improving payload effectiveness and strategy selection for different targets.
* **Develop an interactive dashboard**: A visual interface (e.g., Streamlit or web-based dashboard) could offer real-time monitoring and reporting.
* **Integrate advanced testing tools**: Adding tools such as Burp Suite API, OWASP ZAP, or source code analyzers would enable hybrid (black-box/white-box) testing support.
* **Optimize for performance and cost**: Lighter LLMs (e.g., GPT-4o-mini or open-source alternatives) may be used to reduce latency and operational expenses while maintaining reliability.
* **Enable task planning and dynamic sequencing**: Giving agents the ability to autonomously determine optimal toolchains and task flows based on real-time context.

# ****REFERENCES****

1. Cybersecurity Ventures. (2020). “Cybercrime to cost the world $10.5 trillion annually by 2025.” from <https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/>
2. McCarthy, J. (2007). What is Artificial Intelligence?. Stanford University.
3. **Turing, A. M. (1950).** Computing machinery and intelligence.
4. McCarthy, J., Minsky, M., Rochester, N., & Shannon, C. (1956). Dartmouth Conference Proposal.
5. Haugeland, J. (1985). Artificial Intelligence: The Very Idea.
6. Microsoft. (2025). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. GitHub. <https://github.com/microsoft/autogen>
7. OpenAI. (2025). GPT-4 Technical Report. <https://openai.com/research/gpt-4>
8. Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.
9. SentinelOne. (2025). Purple AI: Accelerating EDR with AI-based threat hunting. Retrieved from <https://www.sentinelone.com>
10. Penetration Testing Execution Standard (PTES). (2022). Retrieved from <http://www.pentest-standard.org/index.php/>
11. CrewAI. GitHub Repository: <https://github.com/joaomdmoura/crewai>