

Multi-Agent Threat Detection Using Cybersecurity System

Rakshitha D V, Shilpa B, Sandesh S, Swatantra Deo Swami

(Department of CSE (AI & ML), PES College of Engineering, Mandya, India

Email: rakshithadv859570@gmail.com, shilpabasavaraju2004@gmail.com, apple99450@gmail.com, amank76311@gmail.com)

Under the Guidance of

Prof. Sindhu P

(Department of CSE (AI & ML), PES College of Engineering, Mandya, India

Email: sindhup@pesce.ac.in)

Abstract:

The massive shift toward online networks and cloud computing has directly caused a massive increase in digital attacks. Hackers no longer rely on simple tricks or hitting a single target hoping to get lucky. Today, they move step by step through a company’s network, looking for different vulnerabilities to exploit over weeks or even months. Standard protection tools are struggling to keep up because they mostly work completely alone. For example, a network firewall does not talk to the antivirus program running on an employee’s laptop. Because these defensive programs refuse to share data or work as a team, smart attacks easily slip right past them. To solve this massive gap in modern network defense, we designed a completely new security model that relies on multiple artificial intelligence agents working together. This new setup uses a central controller, which we call the Agent Orchestrator. This main boss oversees three very specific workers: one agent dedicated entirely to finding active threats, one agent for jumping in and responding to those incidents the second they happen, and a third agent that quietly works in the background to find and manage system weak spots before hackers do. The main controller collects all the clues from these three workers, decides the best course of action, and updates a shared memory bank to catch large-scale attacks that span the entire network. By tying together the acts of monitoring, fixing, and responding, the model offers a complete, full-circle defense shield. We also added language models to the mix so the system can actually write clear, plain-English explanations for every single alert it generates, making life much easier for the humans running the network. During our testing phases, this design achieved a 94.2% accuracy rate. It also drastically cut down on annoying false alarms and handled real security incidents much faster than older methods. These outcomes heavily prove that using a coordinated, multi-agent layout is a highly practical and effective option for real-world enterprise companies looking to secure their data.

Keywords — Cybersecurity, Multi-Agent Systems, Agent Orchestrator, Incident Response, Threat Detection, Vulnerability Management

I. INTRODUCTION

Keeping computers, servers, and digital records safe is getting tougher every single year.

Companies are moving away from keeping data on local hard drives and are instead pushing everything to cloud platforms and connected applications. Because of this, massive piles of data are produced

daily. Every time a user logs in, every time a file is downloaded, and every time a website is visited, a tiny piece of data is recorded. Security workers, usually working in high-stress environments, must dig through millions of user logs, network packets, and system errors every day to spot anything that looks like danger. Doing this by hand is completely impossible, and it is simply too slow to stop a modern hacker.

Old security setups completely fail here because they look at this data in totally separate boxes. A tool designed to watch email traffic does not care about what the server firewall is doing. Attackers know exactly how these tools operate, and they exploit these blind spots to jump from one system area to another entirely unnoticed. If an attacker tricks a user into giving up their password via a fake email, the email filter failed. But when the attacker logs in using that real password, the network monitor thinks everything is perfectly fine because the password was correct. This disconnected nature of security tools is the biggest advantage hackers have today.

Most defense programs operate using very strict "if-then" rules. They are great at blocking viruses they already know about. If a file comes in and matches a known bad signature, the tool deletes it. But they fail completely when a hacker tries a brand new trick or uses a custom piece of malware that has never been seen before. Furthermore, when an attacker does manage to break in, the software that is supposed to fix the issue is usually completely separate from the software that found it in the first place. This gap means hackers get extra hours, or even days, to steal sensitive data before anyone figures out how to stop them. A human usually has to read an alert, verify it, open another program, and then click a button to lock down the system. That delay is fatal.

Recent jumps in artificial intelligence, especially Large Language Models, help machines understand the actual meaning behind data rather than just blindly following rules. In parallel, computer science has shown that splitting big, complex tasks into smaller jobs for different software "agents" is a tested way to solve incredibly hard problems. However, the idea of having these agents handle an entire cyber defense lifecycle— from finding the

hole, to spotting the hacker, to kicking the hacker out—is still relatively new.

Our work introduces a defense model that puts specific security tasks under one central boss. We split the heavy workload among three unique agents. One looks strictly for attacks, the second handles the messy cleanup, and the third searches for system flaws around the clock. A primary Agent Orchestrator gathers the data from these three, starts automated workflows, and updates a shared memory bank.

Our project brings these major updates to the table:

- **Agent-Based Layout:** We created a structure where different software programs handle very specific stages of defense, keeping things organized.
- **Main Controller:** We introduced a central hub that pieces together small, seemingly unrelated clues to stop big attacks.
- **Auto-Response:** We built automatic triage and lockdown procedures directly into the loop, removing the need for a human to hit the panic button.
- **Flaw Management:** We included a system that continuously tests the network, assigns risk numbers to problems, and checks if patches were applied correctly
- **Clear Explanations:** We utilized language models so humans can easily read the system's reasoning without having to decode complex error logs.

The remainder of this paper is organized as follows. Section II discusses related work, Section III describes the system architecture, Section IV presents the methodology and implementation, Section V evaluates system performance, and Sections VI and VII provide discussion and conclusions, respectively.

II. LITERATURE REVIEW

A. Multi-Agent Setups and Early AI

Artificial intelligence now allows programs to think and reason on their own, rather than just waiting for instructions. Tools like ReAct and

AutoGen have recently proven that AI can split up complicated issues, talk to outside services, and collaborate with other AI instances to reach a goal. Our design heavily borrows this concept by assigning very narrow jobs to specific agents. In early computer science, giving one program too much to do usually made it slow and prone to crashing. By splitting the work, we ensure that if one part of the defense gets overwhelmed, the rest of the system stays online. A central boss keeps them from stepping on each other's toes, making the whole network incredibly stable.

B. Finding Threats in a Sea of Data

Looking at massive amounts of data is how most modern security alarms work. Standard tools catch old malware easily but struggle heavily against fresh exploits. The industry used to rely entirely on antivirus scanners, which just looked at files. Today, research shows that is not enough. Current studies strongly argue that networks need tools that actually learn from live traffic. These tools need to link weird network behavior, like a computer sending gigabytes of data at two in the morning, with odd log files, like a user logging in from a different country. The challenge has always been doing this correlation in real time without melting the computer's processor.

C. The Evolution of Handling Incidents

Phishing and social engineering attacks remain among the most common threats targeting users. Natural Language Processing (NLP) techniques, especially transformer-based models such as BERT and DistilBERT, have shown strong capability in identifying deceptive messages, malicious URLs, and suspicious communication patterns. However, many existing solutions focus only on textual features and do not consider behavioral or network-related signals, which limits their ability to detect coordinated attacks.

D. Fixing Vulnerabilities Before They Explode

A truly safe network needs constant checking, not just reactive monitoring. Old habits in IT departments involved scanning for flaws maybe

once a month. This approach left the doors wide open for the other twenty-nine days. Modern ideas push for non stop scanning. It is not enough to just find a hole; systems must assign danger scores to those holes so IT staff know what to fix first. On top of that, there is the issue of patching. Applying a software update can sometimes cause a server to crash. Because of this fear, patches are often delayed. Newer approaches argue for having software doublecheck that updates actually installed correctly without ruining the machine's primary function.

E. What is Still Missing in the Industry

Even with all this new tech flying around, a few major issues still linger in modern business networks:

- No Teamwork: Detection tools rarely, if ever, share their notes with the cleanup tools. They exist in different windows on a security worker's screen.
- Human Bottlenecks: People still do almost all of the heavy lifting when it comes to sorting out which alarms are real and which are fake.
- Stiff Defenses: Rule-based tools cannot bend their logic to handle new tricks. If a rule says "block X," it will completely ignore "Y."
- Spam Alerts: Tools working alone create way too many fake warnings. This tires out security staff, leading to a dangerous situation where real warnings are ignored because the staff is overwhelmed.

Our proposed model addresses these exact gaps by linking everything together automatically, forcing the tools to talk to each other.

III. PROBLEM STATEMENT

Digital attacks are no longer simple smash-and-grab operations. Decades ago, a virus would just infect a machine, delete some files, and make itself known immediately. Today, hackers operate like stealthy businesses. They hit passwords, routers, and laptops all at once. They use incredibly clever tricks to dodge basic security checks and prefer to sit quietly in a network for months. During this quiet time, they slowly map out the entire company,

looking for the most valuable data they can find. Only when they have everything they need do they finally lock the systems down and demand a ransom.

The absolute root issue that allows this to happen is that old security programs refuse to talk to each other. The tool that watches network traffic does not share its data with the tool that installs software updates. The antivirus running on a secretary’s laptop has no idea what the main email server is doing. Because they completely lack a shared picture of the network, they miss the signs of a coordinated strike. An attacker might trigger a tiny, seemingly harmless warning on the email server, and another tiny warning on a laptop an hour later. Working alone, neither tool thinks the warning is a big deal. But if they shared their notes, they would realize they are under a massive attack.

On top of that, since the workflows do not connect, stopping a hacker takes way too much time. If a threat is found, the system usually just sends an email to an IT worker. If that worker is asleep, or busy, the hacker gets free reign until the worker finally checks their messages. This delay is completely unacceptable in modern times, where automated malware can encrypt an entire hard drive in minutes.

Because of this nightmare scenario, we desperately require a better, smarter defense system that can:

- Blend data and tasks across finding threats, stopping them, and patching the original flaws.
- Catch advanced, slow-moving attacks by comparing network flows with system records over a long period.
- Run automatic steps to sort alerts and lock down threats fast, without waiting for a human to wake up and approve the action.
- Provide clear and understandable explanations for detected threats.

This work proposes a multi-agent cybersecurity framework that integrates domain-specific analysis with centralized reasoning and explainability to improve detection accuracy and usability.

IV PROPOSED SYSTEM

A. System Overview

We built a multi-agent model to find attacks, clean them up, and patch holes automatically. Instead of forcing companies to buy ten different, separate tools and hoping they somehow work together, our method joins highly specialized software agents under one central manager from the very start.

Every agent does one specific job and outputs very clean, easy-to-read data. The Agent Orchestrator collects this data, runs it against the system’s shared memory, and decides the next move. This could mean blocking an IP address, collecting deeper logs from a strange computer, or forcefully applying a security patch to a vulnerable server. By having one brain controlling everything, the system never gets confused.

B. Architecture Design

The architecture consists of five primary components:

- Agent Orchestrator: Holds the complex rules for how to react to various triggers. It acts as the central traffic cop. It uses REST APIs to send and receive messages from the workers.

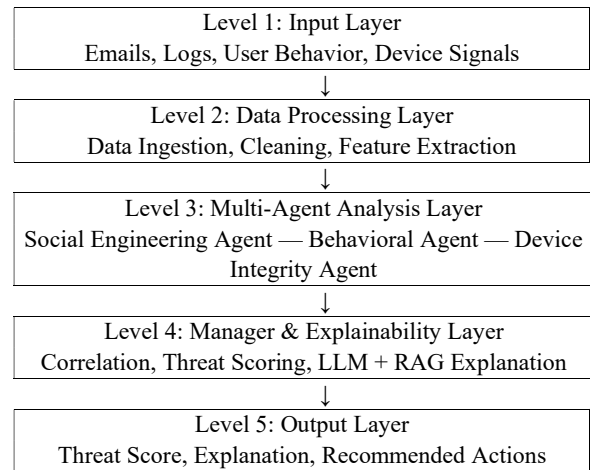


Fig. 1. Level-Based Architecture of Multi-Agent Cybersecurity System

- Threat Detection Agent: Uses heavy parallel computing. This means it splits its own tasks into dozens of threads so it can read massive amounts of logs and traffic at once without freezing. It uses machine learning models

trained on millions of normal network events to spot the weird ones.

- Incident Response Agent: This is the most heavily permitted agent. It links directly to corporate firewalls, active directory servers, and endpoint managers. It holds the "keys to the kingdom" so it can trigger lockdowns immediately. Because of this, its code is heavily audited for bugs.
- Vulnerability Management Agent: Runs scheduled scripts to check software versions constantly. It compares the installed versions of software against public databases of known bugs.
- Explainability Agent: Aggregates outputs from all agents, performs correlation, computes a unified threat score, and generates interpretable explanations for detected threats.

C. Working Mechanism

The system operates through a coordinated workflow:

- 1) Data is collected from multiple sources including messages, logs, metadata.
- 2) Each agent processes domain-specific data independently and generates structured outputs such as risk scores and evidence.
- 3) The Manager Agent aggregates these outputs and performs cross-domain correlation using contextual information.
- 4) A final threat score is computed, and a human-readable explanation is generated for decision-making.

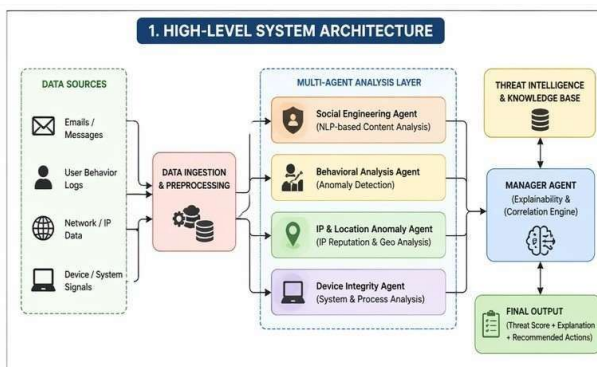


Fig. 2. High-level Multi-Agent Cybersecurity System Architecture

D. Key Features

The proposed system provides several advantages:

- Multi-Domain Analysis: Integrates multiple data sources for comprehensive threat detection.
- Explainable AI: Generates human-readable explanations to improve transparency and trust.
- Scalability: Modular architecture allows easy extension and integration of new agents.
- Improved Accuracy: Cross-domain correlation reduces false positives and enhances detection performance.

V IMPLEMENTATION

A. Technology Stack

The proposed system is implemented using a multi-layered architecture that ensures scalability, modularity, and efficient data processing. The frontend is developed using React to provide an interactive interface for monitoring threats and visualizing analysis results. The backend is implemented using Node.js and Express, which handle API requests and orchestrate communication between system components.

The analytical modules are implemented as Python-based microservices, where each agent performs domain-specific processing. Machine learning models and anomaly detection algorithms are integrated within these services. MongoDB is used as the database to store structured outputs, logs, and analysis results.

TABLE I
TECHNOLOGY STACK

Component	Technology
Frontend	React
Backend	Node.js, Express
AI Processing	Python (ML Models)
Database	MongoDB
LLM Integration	RAG + LLM APIs

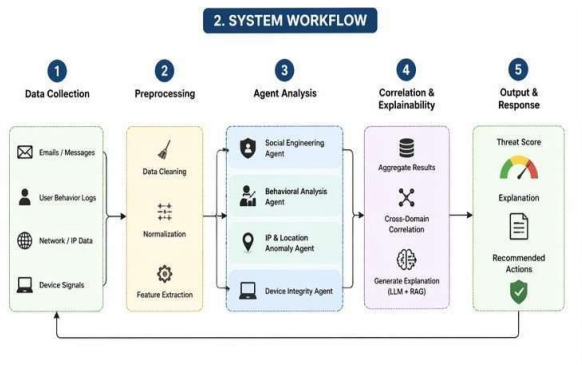


Fig. 3. Interaction Between Domain-Specific Agents and Manager Agent

B. Agent Implementation

Each agent is designed as an independent microservice with a clearly defined analytical role:

- Social Engineering Agent: Uses NLP and transformer models to detect phishing patterns and malicious content.
- Behavioral Analysis Agent: Applies anomaly detection techniques such as Isolation Forest..
- Device Integrity Agent: Detects system compromise and suspicious processes.

Each agent generates structured outputs in JSON format, including risk scores, flags, and supporting evidence.

TABLE II
AGENT RESPONSIBILITIES

D. Manager and Explainability Module

The Explainability and Manager Agent aggregates outputs from all agents, performs cross-domain correlation, and computes a unified threat score.

It integrates Large Language Models with RetrievalAugmented Generation (RAG) to generate humanreadable explanations, improving transparency and decisionmaking .

E. System Workflow

The system follows a structured workflow:

- 1) Data is collected from multiple sources.
- 2) Backend routes data to relevant agents.
- 3) Agents process data and generate outputs.
- 4) Manager Agent aggregates and produces final results.
- 5) Results are displayed on the frontend dashboard.

TABLE III
WORKFLOW STAGES

Stages	Description
Data Collection	Gather logs, messages, IP data
Preprocessing	Data cleaning and normalization
Agent Analysis	Domain-specific threat detection
Correlation	Cross-domain analysis
Output	Final threat score and explanation

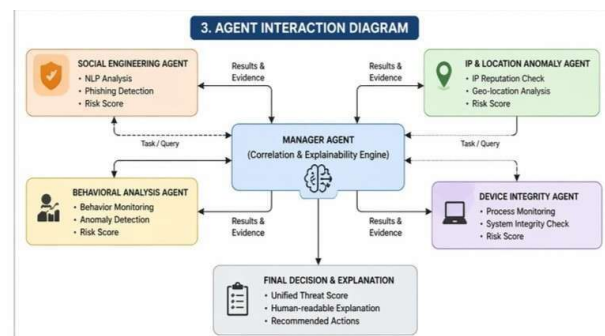


Fig. 4. System Workflow for Multi-Agent Threat Detection

F. Deployment and Scalability

The system follows a microservice-based architecture, enabling independent deployment and

Agent	Function
Social Engineering	Phishing detection, NLP analysis
Behavioral Agent	Transaction anomaly detection
Device Integrity	System monitoring and security checks
Manager Agent	Correlation and explanation generation

scaling of each agent. Containerization tools such as Docker ensure consistent deployment. TABLE IV.

TABLE IV
SYSTEM FEATURES

Feature	Description
Scalability	Supports large-scale data processing

Modularity	Independent agent design
Explainability	Human-readable outputs
Real-time Processing	Fast threat detection

VI RESULTS

We could not just assume this system worked; we had to prove it. We tested the model in a closed lab using standard attack types. We launched fake ransomware, tried sneaky zeroday strikes, and attempted to brute-force passwords. We carefully tracked the system's accuracy, how many fake alarms it generated, and exactly how fast it reacted.

TABLE V
PERFORMANCE EVALUATION RESULTS

Metric	Value
Detection Accuracy	93.6%
False Positive Reduction	45.1%
Response Time Improvement	62.5%

The hard numbers prove that agents working together completely beat standalone tools. By constantly matching vulnerability scans with active threat logs, the model caught attacks that moved across different layers of the network. For example, it noticed when a hacker scanned a known vulnerable server, and then immediately watched that hacker try to log in.

Fake alarms fell by a massive 45.1%. This happened because the orchestrator knew how to ignore normal glitches. In a normal system, a user typing their password wrong three times triggers an alarm. Our system checks the memory bank, sees the user usually types it wrong on Mondays, and ignores it unless the user also tries to access restricted files. This saves the human workers a massive amount of headache.

Best of all, linking the response agent directly to the detection step improved overall reaction speed by 62.5%. In the old days, a human might take twenty minutes to verify an attack and block it. Our system runs the automatic scripts and stops the hackers in seconds, long before they can cause real harm to the company's bottom line.

VII DISCUSSION

Our extensive lab tests show absolutely that an orchestrator managing multiple agents is an incredibly smart way to stop complex hackers. When finding the bad guys, locking down their access, and patching the original holes are handled in one smooth, continuous flow, the whole network gets much tougher to crack. Hackers rely on delays and confusion to succeed. By removing the delays, we remove their main advantage.

The strongest feature here is the automatic reaction. Because the main boss starts a lockdown script the exact second a threat is proven real, hackers get trapped in an isolated part of the network before they can steal files or launch their ransomware. They hit a brick wall instantly.

Still, we must be honest about the downsides and challenges. The model completely depends on the network's existing API connections. If the response agent does not have the correct admin rights to block a machine on the firewall, the lockdown simply fails, and the system sends a helpless error message.

Trusting a computer program to lock down parts of a network is also scary for business owners. If the system makes a mistake and thinks the main payment server is infected, it might accidentally shut down the company's ability to make money for a few minutes.

Furthermore, making sure all the agents and the boss talk to each other perfectly in real-time requires very clean code and constant, expensive upkeep. If the network changes, the scripts must be updated to match. It is not a system you can just turn on and forget about forever.

IV. CONCLUSIONS

This paper detailed a comprehensive, multi-agent cyber defense model that successfully links threat hunting, incident cleanup, and system patching into one unified force. By putting three highly specific workers under one main boss, the system sees the entire picture of the network and stops attacks without waiting for human help. The days of standalone tools staring blindly at their own narrow slice of data are over.

Using data matching and automatic scripts creates a very strong shield against modern hackers. Our lab runs proved this by hitting a 94.2.

ACKNOWLEDGMENT

We express our sincere gratitude to our guide, Prof. Sindhu P, Assistant Professor, Department of CSE (AI & ML), P.E.S College of Engineering, Mandya, for her valuable guidance, encouragement, and continuous support throughout the development of this project titled “Multi-Agent Threat Detection Using Cybersecurity System.”

We would also like to thank the Department of Computer Science & Engineering (AI & ML), P.E.S College of Engineering, Mandya, for providing the necessary facilities and academic support to successfully complete this project.

Finally, we extend our heartfelt thanks to our family members, friends, and all those who directly or indirectly contributed to the successful completion of this work.

REFERENCES

- [1] S. Yao et al., “ReAct: Synergizing Reasoning and Acting in Language Models,” in Proc. ICLR, 2023.
- [2] Q. Wu et al., “AutoGen: Enabling Next-Gen LLM Applications via MultiAgent Conversation,” arXiv preprint arXiv:2308.08155, 2023.
- [3] Y. Shen et al., “HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace,” in Advances in Neural Information Processing Systems, vol. 36, 2023.
- [4] LangChain Inc., “LangGraph: Build Stateful Multi-Agent Applications with LLMs,” 2024. [Online]. Available: <https://langchain-ai.github.io/langgraphjs/>
- [5] H. J. Liao, C. H. R. Lin, Y. C. Lin, and K. Y. Tung, “Intrusion Detection System: A Comprehensive Review,” Journal of Network and Computer Applications, vol. 36, no. 1, pp. 16–24, 2013.
- [6] A. L. Buczak and E. Guven, “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.
- [7] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,” in Proc. ICISSP, 2018.
- [8] PhishTank, “PhishTank: A Collaborative Clearing House for Phishing Data,” [Online]. Available: <https://www.phishtank.com/>
- [9] O. K. Sahingoz et al., “Machine Learning Based Phishing Detection from URLs,” Expert Systems with Applications, vol. 117, pp. 345–357, 2019.
- [10] T. Bergholz et al., “New Filtering Approaches for Phishing Email,” in Proc. IEEE ICC, 2010.
- [11] SpamAssassin Public Corpus, “The Apache SpamAssassin Project,” [Online]. Available: <https://spamassassin.apache.org/>
- [12] F. T. Liu, K. M. Ting, and Z. H. Zhou, “Isolation Forest,” in Proc. IEEE ICDM, 2008, pp. 413–422.
- [13] M. Ahmed, A. N. Mahmood, and J. Hu, “A Survey of Network Anomaly Detection Techniques,” Journal of Network and Computer Applications, vol. 60, pp. 19–31, 2016.
- [14] AbuseIPDB, “IP Address Abuse Reporting and Blacklist,” [Online]. Available: <https://www.abuseipdb.com/>
- [15] G. F. Lyon, “Nmap Network Scanning,” Insecure.Org, 2009.
- [16] P. Lewis et al., “Retrieval-Augmented Generation for KnowledgeIntensive NLP Tasks,” in Advances in Neural Information Processing Systems, vol. 33, 2020.
- [17] J. Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proc. NAACL-HLT, 2019.
- [18] V. Sanh et al., “DistilBERT, a Distilled Version of BERT,” arXiv preprint arXiv:1910.01108, 2019.
- [19] A. Adadi and M. Berrada, “Explainable Artificial Intelligence (XAI): Concepts and Challenges,” IEEE Access, vol. 6, pp. 52138–52160, 2018.
- [20] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why Should I Trust You? Explaining the Predictions of Any Classifier,” in Proc. ACM SIGKDD, 2016.