StuxSwarm: A Threat Analysis of Fully Autonomous AI-Orchestrated Cyber Attacks

Liberato Aguilar
Department of Computer Science and Engineering
Texas A&M University
College Station, TX, USA
liberatoaguilar@tamu.edu
UIN: 730001728

Course: CSCE 765 — Network Security
Instructor: Dr. Hamilton
Spring 2026

***Abstract*—Historically, cyber attacks between countries have been year-long efforts between multiple contributing intelligence agencies that require gathering and sharing intel from many different attack angles so the final outcome is successful. One very famous example of this is Stuxnet, which was a co-effort from at least five different countries, with the main actors being the United States and Israel. This entire effort took years to plan, but with the advent of AI, is it possible to make this style of attack repeatable at scale? Most famously, large language models (LLMs) have achieved great coding performance, already surpassing the average junior software engineer in many cases. We know that there are ways to uncensor or "jailbreak" LLMs and use their intelligence for malicious purposes, such as writing malware. Other AI models include new photorealistic generators such as Google Nano Banana or the new generation of video generation models such as SeedDance 2.0. Voice cloning models are now increasingly used for phishing and ransom scams. It is reported that 76% of phishing campaigns today use some form of AI [14]. These models allow for generating artificial scenes featuring any given human subject as input. Today these have mainly been used for pranks amongst friends, and major companies claim to watermark their outputs with invisible signatures to mitigate malicious use. The question many attackers are now asking is whether it is now feasible to create an end-to-end automated cyber attack framework that leverages the ever-decreasing cost of AI models with no censorship to exponentially increase the amount of attacks by lowering the monetary and time cost of setup per attack. This paper explores what a novel attack framework, which we call StuxSwarm, could look like and what should be done to defend against it.**

***Index Terms*—Stuxnet, StuxSwarm, artificial intelligence, large language models, autonomous cyber attacks, social engineering, deepfake, malware, digital forensics**

I. Introduction

Stuxnet combined methods of social engineering, zero-day exploits, hardware vulnerabilities, and proper malware engineering to get the final outcome of severely hindering Iran's nuclear capacity by decades of advancement [5]–[7]. The main attack layer started with blackmailing a repair contractor for Siemens, who manufactured the centrifuges and the PLCs that Iran was using to refine uranium for nuclear warheads. Keeping hostages is a common way to manipulate workers inside the target facility for easy access to internal IoT devices. Knowledge of zero-day exploits could require some contribution from the manufacturer, but these attacks could also be discovered without notifying anyone. Improper hardware security such as open access to USB ports is another clever way to inject malware into the entire system [7]. Developing the worm itself that lived inside the USB was not easy, as it had to be designed to only target Iran's computers and self-delete after the attack was completed. Overall the attack took years to set up and execute, and even then there was a bug which caused the worm to not fully delete itself, ultimately leading to the discovery of the attack. Next we explore how a modern recreation of Stuxnet would look and how long it would take to execute.

State-of-the-art LLMs such as GPT-5.3 Codex or Claude Opus 4.6 excel at coding, and many claim that there is no longer any need for manual coding other than for small tweaks [1]. LLMs are also much faster at producing great code compared to humans. The Stuxnet worm was about 0.5 MB in size (˜15,000 lines of code), which is equivalent to 400,000 tokens, something that any state-of-the-art LLM can write in minutes. Debugging the program would take longer, but it is now feasible to have a working worm in a few hours, not months. LLMs today not only have the ability to code, but also reason, and there is much research into having LLMs autonomously pen-test hardware devices to find and patch zero-day exploits [1], [3]. Any malicious actor could use this same approach to have an agent look for zero-day exploits 24/7 and build an attack once a serious flaw is found. The social engineering aspect could bypass actually having to kidnap a hostage and just use any images taken of the "hostage" to generate a believable set of photos and videos in a few hours to convince a worker inside the target location to grant access [8]. This already covers the most common pipeline of attacks done today, and it is not difficult for an attacker to leverage these technologies to build and execute attacks autonomously.

The remainder of this paper is divided into seven sections. Section II gives a technical background on the Stuxnet worm and what it took to build and execute the attack. This gives a natural transition into Section III, which explores whether modern AI capabilities can replace human roles in such an attack. Section IV formalizes the agentic architecture—which we name StuxSwarm—into something an attacker could build and scale. Section V details how feasible the scalability

actually is and how dangerous such a system could become. Section VI presents GuardSwarm, a defensive agentic system designed to combat this new generation of malware attacks. Section VII discusses limitations, and Section VIII concludes.

II. Background: The Original Stuxnet

A. Attack Overview

The Stuxnet attack was a collaborative effort between multiple countries, namely the United States and Israel [5], [6]. This attack is historical because it was unlike any other previous malware up to that point. Details of the attack were first uncovered by the Belarusian cybersecurity company VirusBlokAda [7]. The main differentiator was the level of complexity, collaboration, and knowledge needed to design and deploy it. Stuxnet was the first malware designed to specifically target industrial control systems (ICS) and cause significant real-world damage to Iran's nuclear enrichment capabilities. The primary target was the Siemens Simatic S7-300 and S7-400 PLCs, which controlled centrifuges at the Natanz uranium enrichment plant. The software itself traveled through three tiers: PG/HMI (specifically Windows PCs), PLCs, and the centrifuge field devices themselves. It was designed to be lightweight and undetectable, taking up only ˜0.5 MB to ˜1.2 MB of storage [7].

B. Attack Phases

The first phase of Stuxnet begins with three different infection vectors. The first is via an infected USB memory stick connected to a Windows PC. Early versions of Stuxnet relied on a built-in functionality of USB sticks that allows auto-running executable software via an `autorun.inf` configuration file. Later versions exploited the Microsoft Windows Shortcut 'LNK/PIF' Files Automatic File Execution zero-day vulnerability (CVE-2010-2568). The second infection method copies itself onto other devices on the local network by exploiting one of two zero-day vulnerabilities: the Microsoft Windows Print Spooler Service Remote Code Execution vulnerability (CVE-2010-2729) or the Microsoft Windows Server Service RPC Handling Remote Code Execution vulnerability (CVE-2008-4250) [7]. This cleverly creates a malicious print request to printers connected on the local network which contains arbitrarily executable code. Other network replication methods included copying itself to shared folders on the network or onto WinCC database servers. Interestingly, Stuxnet was designed to only use the local network to spread if the date was before June 1, 2011. The third attack vector hides itself in STEP 7 folders, infecting other computers when project files are shared between PCs. This third vector was not fully completed according to later analysis. In total, the infection phase used at least three zero-day attacks, which was unheard of at the time for malware exploits [7].

After infection, Stuxnet exploited two additional zero-day vulnerabilities: the Windows Task Scheduler Privilege Escalation vulnerability (CVE-2010-3888) and the Windows Win32K Keyboard Layout vulnerability (CVE-2010-2743). These allowed Stuxnet to raise its own privileges on the infected host machine. With root privileges, Stuxnet installed two drivers at the hardware level, designed to be undetectable so as not to raise red flags with antivirus software. These executables are where Stuxnet gets its name: `mrxnet.sys` and `mrxcls.sys`. Both drivers were signed with stolen private keys from Realtek and JMicron, so the Windows OS ran them with kernel privileges on startup without issue [7].

Table I summarizes the six vulnerabilities exploited by Stuxnet.

TABLE I
VULNERABILITIES EXPLOITED BY STUXNET [7].

#	CVE	0-day	Description
1	CVE-2008-4250	No	Windows RPC Remote Code Exec.
2	CVE-2010-2568	Yes	LNK/PIF Auto File Execution
3	CVE-2010-2729	Yes	Print Spooler Remote Code Exec.
4	CVE-2010-2743	Yes	Win32K Keyboard Priv. Escalation
5	CVE-2010-2772	Yes	Siemens WinCC Default Password
6	CVE-2010-3888	Yes	Task Scheduler Priv. Escalation

Kernel-level rootkit exploits gave Stuxnet full power over the host machine, and the real damage could begin. Stuxnet installed an RPC server to move itself laterally through the local network and install itself on more machines, repeating the steps highlighted above. Since all infected machines could interact with each other via the RPC server, this allowed every infected machine to check what version of Stuxnet it was running and update itself if a neighbor had a newer version [7].

Stuxnet could theoretically be installed on any Windows device, but it only acted on devices that met very specific criteria found only in Iranian nuclear enrichment centers. To scout for these parameters, the worm first looked for the Siemens WinCC software on the host machine. This software is only installed on machines used to control industrial PLCs, which send commands to physical hardware like the centrifuges. Once detected, Stuxnet would overwrite the `.DLL` used by the WinCC system with its own modified version. This `.DLL` included changes and additional instructions to many read and write functions the PLC commonly used. The strategy is comparable to a man-in-the-middle attack, where the new `.DLL` would intercept data and relay different data back and forth. Once access to the PLC was secured, Stuxnet attempted to connect to remote HTTP servers located at `www.mypremierfutbol.com` and `www.todaysfutbol.com`, which were non-suspicious URLs designed to pass through firewalls, where it would send all device information about the infected host [7].

With full flexibility to send commands to the centrifuges via the infected PLC, Stuxnet would constantly vary the spin frequency, ranging from 1410 Hz, down to 2 Hz, and back to 1064 Hz [7]. This rapid change in frequency would both damage the hardware and lead to poor enrichment results, all while reporting back to the Windows HMI that all systems were working perfectly. This style of PLC rootkit exploit, designed to cause real physical damage, had never been used before and marks a turning point in malware history. After the

damage was done, Stuxnet was programmed to delete itself from the host on June 24, 2012, removing any trace that it was ever there [7].

This was a brief high-level overview of Stuxnet. In reality there is much more to explain about the details of implementation, much of which has never been officially made public. Many governments recruited top cybersecurity talent to reverse-engineer as much as possible, but many details will likely remain unknown for a long time.

C. Manpower Required

There have been many independent estimates on the exact manpower and time required to develop Stuxnet, and the general consensus lies around 45 people with tasks ranging from development to on-site installation of the USB memory cards [7]. Table II shows the complete breakdown according to the Falco report.

TABLE II
ESTIMATION OF PERSONNEL NEEDED FOR STUXNET DEVELOPMENT, DEPLOYMENT, AND OPERATIONAL MANAGEMENT [7].

Profile	Number
ICS consultant	2
SCADA/PLC architect/engineer	2
Simatic PLC programmer	3
Nuclear fuel production expert	2
Windows internal system programmer	5–10
Vulnerability analyst	2
Exploit writer	3
Quality assurance operator	3
Lab field tester	3
IT/C&C infrastructure maintainer	5
On-site intelligence operator	1–10?
On-site installer	1–2?
TOTAL	**≈45**

Notice the wide range of skills that all came together to complete such a complex worm attack. First, the experts on the target hardware devices were needed. The ICS consultants, SCADA/PLC architects and engineers, and Simatic PLC programmers all knew how the target machines worked and what specific commands Stuxnet would need to send to change the frequencies of the centrifuges. An expert with background knowledge on nuclear fuel enrichment was needed in the first place to know what parameters and modifications to the system would be most destructive while staying undetectable. For infecting host machines running Windows, many Windows systems programmers were crucial. This group was probably tasked with writing code related to hardware drivers, USB programs, printers, and servers in Windows. In conjunction with the Windows programmers, the actual exploit writers and vulnerability experts knew how to get root access and elevate permissions when arbitrary code execution was needed. For testing the worm before deployment, the lab field testers came in. Maintaining the servers where Stuxnet sent over the host machine's information required a dedicated IT and C&C team. Installing Stuxnet required manual labor, so there was a need for on-site operators and installers who were most likely recruited or coerced into plugging in the infected USB and starting the attack [7].

The full attack was estimated to take many months and up to two years to fully develop, maintain, and update [5]. Multiple countries are suspected of collaborating to complete the attack. Full knowledge of Siemens machines also implies that the manufacturer might have contributed by providing some zero-day exploits before patching them later. Despite recruiting the top specialized talent from multiple nation-states, the software in the worm still contained a few bugs that eventually led to its detection. Some third parties suspect that differing levels of expertise in software development led to common mistakes such as weak encryption in some areas and strong encryption in others, unfinished code blocks (STEP 7 exploits), and inflated executable files that could be detected by antivirus software [7].

III. AI Capabilities Enabling Autonomous Attacks

With the advent of artificial intelligence, it is easier than ever to use AI for malicious purposes. There are a reported 900 million weekly active users of ChatGPT alone as of early 2026, and AI has been reported to speed up writing code by up to 55% compared to manually writing it, according to a controlled study of 4,800 developers using GitHub Copilot [1]. Image and video generation models have become ultrarealistic, with newer models maintaining character consistency from a single source image [16]. The LLM space has also seen a growth in model harnesses, which expose a base model to tools such as file I/O, code execution, web search, and bash terminals, giving the model full control of a host computer. The first popular harness was introduced by Anthropic in 2024, named Claude Code. Since then, multiple competitors have been developed, the most popular being the open-source project OpenClaw. Harnesses are now model-agnostic, can spawn subagents, and have full autonomy powered by cron-based heartbeats.

Despite large AI corporations promising ethical development practices, there are many smaller independent labs that do not abide by the same rules [3]. Many open-source models are unfiltered and unaligned, attracting malicious parties willing to trade off some model performance for fewer restrictions. The remainder of this section explores how an attacker could develop a harness-based agentic architecture with multiple subagents leveraging specific tools for malicious intent.

A. Zero-Day Discovery

Stuxnet's main operating mechanism relied on six total zero-day exploits, which most likely took years to find and develop. Only a team of very elite researchers with decades of knowledge would have had the required expertise for an attack like Stuxnet. Less than two decades later, the rise of AI has dramatically lowered the time and cost of finding zero-day exploits. Anthropic recently unveiled Project Glasswing, a collaboration between at least 12 large companies and OS developers including Apple, Microsoft, Google, and the Linux Foundation [13]. This project uses a new class of model called

Claude Mythos, which outperforms any models released to the public. Anthropic's own system card states that AI "has reached a level of coding capability where [it] can surpass all but the most skilled humans at finding and exploiting software vulnerabilities" [17]. This model achieves 83.1% on CyberGym and 93.9% on SWE-bench Verified, compared to Anthropic's latest public model, Opus 4.6, which achieved 66.6% and 80.8% respectively [17]. Mythos autonomously found zero-day vulnerabilities in every major OS and browser, including a 27-year-old vulnerability in OpenBSD—known for being the most security-hardened OS—that had never been identified by humans. It also autonomously chained Linux kernel exploits for full privilege escalation and achieved complete control over the OS [13]. Anthropic claims these vulnerabilities were found autonomously by the model with no human steering at all. Mythos was considered too dangerous for Anthropic to release publicly, and access is restricted to partner companies and the U.S. government as of this writing [17].

Despite Mythos not being released publicly, it is only a matter of time before open-source LLM developers improve their models to reach a similar level. As inference costs keep falling, it is reasonable to expect a Mythos-level model available to all within a few years if not sooner. Giving the public access to such a powerful automated exploit-finder is potentially very dangerous and will surely be used for malicious applications. Anthropic's decision to keep the model closed and give companies a head start on patching their vulnerabilities before a public equivalent model appears is a prudent defensive strategy [13]. OpenAI similarly had concerns about malicious use when GPT-2 was previewed and decided not to release it at first, even though by 2026 GPT-2 is considered a legacy model.

B. Malware Generation via Jailbroken LLMs

Even if Anthropic decides to release a version of Mythos to the public, it will be severely fine-tuned and restricted to detect malicious intent on the tasks assigned. Most public LLM providers have restrictions on what types of questions can be answered for safety purposes, including OpenAI, Google, and Anthropic. Even from the very first releases of ChatGPT, users quickly found that certain prompts "trick" the LLM into responding when it normally would not be allowed to [4]. Now "jailbreak" prompts are more sophisticated, and even more powerful are completely retrained models that remove built-in restrictions and use dark web data. Two notable examples include FraudGPT and WormGPT, which are chatbots specifically designed to be malicious [3]. Expanding on unrestricted chatbots, it is possible to leverage their relaxed permissions as part of a polymorphic virus system like PROMPTFLUX or PROMPTSTEAL, both of which self-modified their code using LLMs during execution [8], [9]. Given the rapid token output rates of modern LLMs, it is reasonable to expect a 400K-token program like Stuxnet could be generated in minutes. This does not include the prior research of finding the exploits or ironing out bugs, but even then it is reasonable to assume polishing the code could take weeks instead of years.

C. AI-Powered Social Engineering

One of the key aspects of Stuxnet was the social engineering and human intelligence needed to gain access inside the actual physical nuclear plant and install the worm via USB [7]. Some suspect that this was achieved via blackmail or hostage strategies. Government agencies are known to conduct intelligence reconnaissance operations such as finding a person's social media and gathering information about their personal lives and their families and friends [12]. From this, lateral targets important to the main target can be used to convince a person to participate in deploying the attack. Today many of these social engineering attacks manifest as scam calls that are easily ignored by many tech-savvy individuals. The effectiveness of these attacks has declined as it has become very popular for YouTube channels to play along with scams and waste the attacker's time for entertainment purposes.

In reality, these types of scams are still prevalent, and scam call centers only need one victim to make it worth their time to keep repeating the scam. Nowadays, with modern AI tools, social engineering methods have become more convincing than ever [12]. OSINT reconnaissance tools such as Maigret, GHunt, and Social Analyzer allow for automated full online profiling of potential victims. Maigret searches 3,000+ websites for a given username and extracts any personal information including names, locations, and even photos, working recursively with any new profiles found. GHunt leverages Google account information from a given email. The Social Analyzer tool is similar to Maigret, searching 1,000 websites for multiple variations of a username. These OSINT tools can be chained and used with other tools like theHarvester to map an organization's infrastructure and gain more information on how to format a potential attack. With basic profile information it could be sufficient to socially manipulate someone for monetary gain, but the real risk comes with publicly available photos and videos of the victims.

State-of-the-art photorealistic generation models can take any image of a person and use it as a starting point and make any edits a user may want, essentially serving as an instant Photoshop. Even though companies like Google are very careful about what type of content their models can generate, other open-source image models are not far behind in quality and have fewer restrictions. From there a false image can easily be passed into a video generation model to create very realistic videos with audio included. ElevenLabs and other leading text-to-speech AI producers have made voice cloning quick and simple, requiring only a short audio sample of a person's voice to produce a synthetic version that can say any phrase. These examples may seem exaggerated in practical use, but there are several real cases of malicious applications and legal consequences. In early 2024, the UK engineering firm Arup lost $25 million after an employee was tricked by an AI-generated video conference featuring a deepfake of the company's CFO and other executives [15]. In 2019, a UK

energy firm was defrauded of $243,000 when a CEO's voice was cloned to authorize a fraudulent wire transfer [18]. It seems attackers are increasingly experimenting with this style of attack, with deepfake-enabled vishing surging 1,600% in Q1 2025 [14].

D. Agentic Orchestration

With the modern capabilities in zero-day exploit finding, malware code generation, and social engineering, it definitely makes it easier for the common attacker to design a cyber attack, but it still requires much manual labor. The attacker still has to guide the AI model to find zero-day exploits. Once an exploit is found, the attacker must guide an AI model to write specific worm code to handle the replication, over-the-air updates, telemetry, and self-destruction. Not to mention the manual research and specific domain knowledge such as how nuclear enrichment works in Stuxnet's case. Social engineering aspects may make it easier to gather a profile and create convincing deepfake footage, but a human must still be in the loop to stitch all the tools together. Even with the best tools it is up to the skill of the attacker to actually make the attack profit worth the effort put in. This is why many cyber attacks are not sophisticated, single precise-target attacks but rather many simple attacks hoping a good percentage of them are profitable. This is why many spam emails are not personalized and scam calls depend on making thousands of calls hoping one conversation turns a profit. Most cyber attackers cast a large fishing net for small fish, rather than hunting a huge whale.

As agentic workflows become more common and harnesses like OpenClaw expose more tools and integrations to increasingly intelligent large language models, much of the human orchestration can now be automated. The entire stack of tasks that a human must perform can be achieved in a much shorter amount of time using the default tools available to the agent along with special instructions on how to achieve certain tasks. The agent has access to basic file read and write, enabling it to produce code in any programming language. After writing code, the agent has the capability to execute and test the newly written code using the code execution tool, specifically useful for testing zero-day exploit capabilities when using a model like Claude Mythos. Although LLMs are very knowledgeable, hallucination is a known problem mostly solved via grounding with web search. Agents have a web search tool that allows them to gather up-to-date information on any topic and consolidate intelligence reports or crawl social media accounts. Native image and video generation are tools available via OpenClaw and support using source images to edit or generate videos from them. There is a text-to-speech integration that supports multiple APIs including ElevenLabs, which was previously shown to support instant and professional voice cloning. Along with code execution, the agent has access to a full bash terminal and an elevated permissions mode allowing the agent full sudo root user access. The bash terminal could be the most dangerous tool available, as it has full knowledge of how to leverage the suite of hacking CLI tools already available (e.g., the Kali Linux suite). To make an agent run autonomously, the harness supports cron jobs, or heartbeats, which "wake" the idle agent and prompt it to complete any remaining tasks. The heartbeat can be scheduled to run every hour and scan for new vulnerabilities, monitor social media accounts, or retry failed exploit attempts. Agents never get "tired" and can run 24/7 autonomously, only updating the human attacker once something important happens (such as receiving a ransom payment). All these tools already exist today; no novel technology needs to be developed by an attacker and only requires one-time setup by a human to continue running. Since the OpenClaw harness is model-agnostic, any LLM, image, video, and text-to-speech model can be interchanged as new, more capable models are released. Multiple models can be used for different tasks to leverage specific strengths of certain models over others.

Another key strength that the harness provides is the ability to spawn and manage subagents, which delegate and parallelize specific tasks to other agents who have the full suite of tools mentioned above. This makes attacks a coordinated agent effort, more similarly mirroring Stuxnet's distributed team of developers and maintainers. The difference between human teams and agent teams is that agents are far more efficient at coordinating efforts and work much faster, longer, and for a lower cost than humans. This greatly accelerates the potential of developing not only a single precise attack, but potentially hundreds or thousands of precise attacks in parallel. As cyber attackers nowadays cast large fishing nets for small fish, an agentic workflow is more analogous to assigning an entire team of hunters for every whale on Earth. The next section explores what such a multi-agent architecture might look like and what tasks each subagent would handle for a Stuxnet-style attack.

IV. Proposed Multi-Agent Attack Architecture: StuxSwarm

We name this architecture **StuxSwarm**, combining the "stux" prefix from the original Stuxnet naming convention (derived from `.stub` files and `mrxnet.sys`) with "swarm" to reflect the multi-agent parallel nature of the proposed system. OpenClaw supports defining subagents with specific instructions using simple markdown files that are appended to a model's system prompt. This makes it easier than ever to define custom agents with specific tasks and goals, creating a hierarchy of agents similar to what we already see in regular development teams. In the following subsections we map many of the jobs Stuxnet required to five separate subagents who collaborate together to achieve the full automated attack workflow. One thing to note is that depending on the style of attack the human attacker wants to create, multiple instances of some agents may be needed or some might not be needed at all. The conductor agent will decide which agents are necessary to complete the job, while the human just has to describe the high-level overview of the attack they want to conduct.

A. Agent 1: Reconnaissance Agent

The first phase of every attack begins with reconnaissance: gathering intel on potential victims and compiling relevant research needed for the specified attack type. This agent uses the web search tool to create a broad list of potential victims, given specific criteria by the main conductor agent or the human. The parameters to search for will vary for the style of attack, and to keep consistency with Stuxnet, some good parameters to search for would be people who work at Siemens as field operators in a certain city. This already vastly narrows down the search pool to very specific individuals. If the search results yield no results, the agent broadens the parameters automatically. Once a set of usernames is compiled, this agent runs Maigret via the bash tool to find all relevant social media accounts and extract personal information, photos, and videos publicly available. From there, other tools like theHarvester map the victim's target organization and gather domain information, emails, org charts, and other employees. In Stuxnet's case we are looking for coworkers who could also be victims. From the Falco report, this agent maps to the 1–10 on-site intelligence operators and 1–2 on-site installers [7]. The final output is a set of directories for each potential victim that includes their social profiles, personal relations, and org charts.

The second function of the reconnaissance agent is to gather up-to-date information on the specific domain knowledge needed to execute the attack. This is analogous to Stuxnet's nuclear fuel production expert and ICS consultants. Given the attack vector designed by a human or the conductor agent, another reconnaissance agent is launched to compile a database of relevant information that will be available to all other agents for more context on the specific attack type. In Stuxnet's case it would be nuclear enrichment documents, while for a general scam case it could be general information about the victim company's area of business.

Reconnaissance agents linked to a specific attack become idle once they are no longer necessary and the information has been compiled. They may be reactivated by the main conductor agent if more victims are needed or if more general information is required. In general this agent will not require a heartbeat or run 24/7.

B. Agent 2: Exploit Discovery Agent

Project Glasswing using Claude Mythos already implements some version of an exploit discovery agent [13]. This agent runs fully autonomously 24/7 until it successfully finds and executes a zero-day exploit on the target machine. For Stuxnet-style attacks, this agent will be tasked with finding exploits for Windows that allow arbitrary code execution and privilege escalation, as well as for Siemens PLC devices. This agent replaces the Windows internal system programmers, vulnerability analysts, and exploit writers from Table II. One note is that a physical Siemens PLC would be needed by an attacker for the agent to test with. For lower-level attacks, access to a PC would be enough to achieve simple monetary ransom-style scams. This agent can also rely on research found by the reconnaissance agent to help it progress much faster. This agent does not write the entire worm software; it only outputs proof-of-concept scripts that another agent can use as reference. Since finding exploits could be a long-running task, OpenClaw's heartbeat feature ensures the agent keeps running until the list of exploits is found and the test scripts are written.

C. Agent 3: Weaponization Agent

The weaponization agent is launched by the conductor once it has verified that the exploit discovery agent is done. With the proof-of-concept scripts, this agent—powered by a jailbroken LLM—writes the rest of the worm code, integrating the newly found zero-day exploits [4]. This agent has full read/write access along with code execution to test the worm and debug the program until it works, which is verified by the conductor. It is important for this agent to understand the requirements of writing malware such as stealth (not detectable by antivirus), obfuscation, polymorphism, encryption, replication, and self-destruction [2]. The output from the reconnaissance agent will also be valuable for writing malicious code because this agent can make specific assumptions from the research gathered and modify code or payload as needed (specific PLCs, OS versions, etc.). Polymorphism is particularly interesting because the worm can embed LLM calls at runtime to self-modify, similar to the PROMPTFLUX approach [8]. Once the code is ready to be executed, the executable is packaged and made available to the conductor agent.

D. Agent 4: Social Engineering Agent

Once the full profile of potential victims and target organization information is compiled by the reconnaissance agent, the social engineering agent takes over to make contact with the victims and manipulate them to continue the attack. The primary tools used by this agent are image and video generation, text-to-speech, web search, file I/O, and OpenClaw's built-in gateways that give it access to common messaging platforms. LLMs have the ability to curate custom manipulation strategies to best exploit a person's identified weaknesses [12]. This could mean creating convincing deepfake videos using reference images to simulate hostage situations or pretending to be a person the victim knows via voice cloning and making fake phone calls. The ultimate goal of this agent is to continue conversations and provide false media until the victim is convinced to either make a fraudulent monetary transfer (like the Arup attack [15]), or plant the malware developed by the weaponization agent (like Stuxnet). In the case of planting a worm, this agent collaborates with the conductor to confirm the malware has been activated properly via telemetry updates from the infected machine.

E. Agent 5: Conductor Agent

The conductor agent is the main supervisor of all other agents and has direct contact with the human attacker. This agent takes high-level information on the style of attack the human wants to conduct and devises a detailed strategy plan with assigned tasks, determining how many of each subagent will be required to complete the full end-to-end attack. Since

not all attacks are the same, this agent must be available 24/7 to make manual corrections and pivots if needed to facilitate the flow and coordination of each agent. Certain agents can work in parallel, and the conductor pieces together the outputs and launches dependent agents when they are needed. For example, in a Stuxnet-style attack, the conductor would launch a few reconnaissance agents and a few exploit discovery agents to begin. These agents periodically report their progress back to the conductor, which relays information between them if necessary. Once the conductor verifies the initial agents are done, it launches a weaponization agent and a few social engineering agents to complete the attack. The conductor has full control over the agents and can ensure they continue working until the end criteria is met. Once the final task has been completed, the human attacker is notified.

F. Summary and Comparison

Every tool and AI agent architecture described above exists today, and the only missing piece is creating the right agent files, instructions, and model integrations that make the full system work. Table III compares the original Stuxnet effort with the proposed agentic architecture.

TABLE III
COMPARISON OF ORIGINAL STUXNET VS. THE PROPOSED STUXSWARM ARCHITECTURE.

Factor	**Stuxnet (2010)**	**StuxSwarm**
Personnel	≈45 specialists	1 human operator
Dev. time	Months to years	Days to weeks
Cost	Millions (est.)	Thousands
Targets	1 (Natanz)	N in parallel
Expertise	Nation-state level	Medium technical
24/7 operation	No (human limits)	Yes (cron/heartbeat)
Scalability	Not scalable	Highly scalable

Figure 1 illustrates the proposed multi-agent architecture and the data flow between agents.

Stuxnet took years to develop with at least ˜45 people, while one human could wire up this system in a few weeks and let it run for another few weeks while monitoring progress hands-off. The human could go about their daily life while the agents do the work. A full nation-state attack could be more complex to orchestrate using a fully agentic system, but common phone scams could get much more sophisticated and automated while targeting regular citizens. The reach and potential upside is much greater using an automated system like this, and it does not require extensive initial setup or deep knowledge about exploits or even software development. This opens the door to many more criminals, so it is more important than ever to build robust protections against agentic cyber crimes, a topic detailed in Section VI.

V. SCALABILITY ANALYSIS

Before the age of AI, multi-step cyber attacks on the level of Stuxnet were not easy to coordinate and took years of effort from many countries to execute. Even then it was not perfect, and the worm was eventually discovered. The StuxSwarm architecture described in the previous section could be set up and maintained by a single attacker with medium-level technical knowledge. Since the architecture is much more efficient and cheaper to run compared to large human teams, attackers could reallocate their budget to run N instances of the system in parallel, attacking at least N targets simultaneously. The main disadvantage of running multiple instances is the cost of running multiple AI models 24/7. API costs and rate limits will slow down the attacks, and if the attacks themselves are not successful, it is only a matter of time before the human attacker either scales down or discards the system. This might seem hopeful, but API costs per token for LLMs have been dropping rapidly over time, and many open-source models can be run entirely locally given a powerful enough machine. The OpenClaw harness supports fallback models, so even switching models when a rate or budget limit is encountered can be automated.

CrowdStrike, one of the main partners of Anthropic's Project Glasswing, stated that "the window between a vulnerability being discovered and being exploited has collapsed—what once took months now happens in minutes with AI" [13]. Their metrics are also very concerning: they observed an 89% surge in AI-enabled attacks year-over-year, and a 65% decrease in breakout time for eCrime, down to just 29 minutes [14]. These are real concerns from large companies in the space, and attackers have been taking advantage of AI. New Malware-as-a-Service kits have been made available for very cheap prices, some as low as $50 [14]. A modern analysis of phishing campaigns found that 76% now use some sort of AI-generated content [14]. The economics of cyber attacks has fundamentally changed, and attacks that once required a nation's budget can now be completed by single individuals or small teams.

Although attacks have been increasing in frequency and specificity, not all hope is lost. Most attacks in 2025 (82%) were still malware-free attacks that focused on the social engineering aspect, not installing malicious software on devices [14]. There is an eternal cat-and-mouse game between attackers and companies patching their system vulnerabilities, and this gap has not been totally surpassed by attackers. It is still difficult to find zero-day exploits, but as Anthropic demonstrated with Claude Mythos, the gap is closing quickly. Anthropic's own system card warns: "we find it alarming that the world looks on track to proceed rapidly to developing superhuman systems without stronger mechanisms in place" [17]. If the developer of Mythos believes that it is too dangerous to be released, companies must move quickly to use the same tools the attackers would and patch their own systems. It is more important than ever for companies to implement stronger mechanisms to defend against AI-enabled attacks, because criminals will begin targeting small companies that decide not to enhance their security instead of going for single individuals or bigger corporations. As NVIDIA CEO Jensen Huang argued on *The Joe Rogan Experience*, the same AI technology used for attacks will also power the defense: "my AI is going to take care of me…you use the same AI

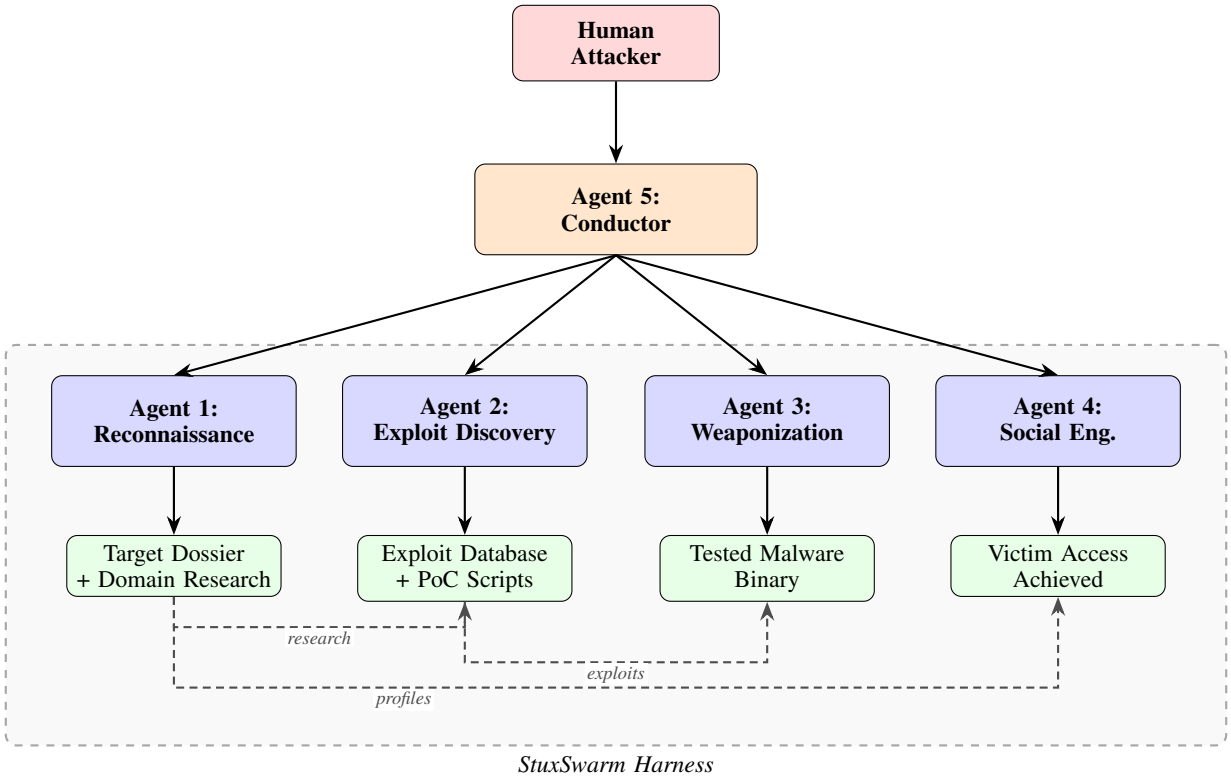

Fig. 1. StuxSwarm: proposed multi-agent attack architecture. Solid arrows indicate command and control flow from the conductor. Dashed arrows indicate data sharing between agents. All agents operate within the OpenClaw harness.

technology to go defend against it" [19]. The best defense against StuxSwarm is its mirror: an agentic cyber defense system we call **GuardSwarm**.

VI. Defensive Framework: GuardSwarm

Just as it has become very accessible to deploy sophisticated attacks, cyber defense teams and white-hat hackers have access to the same tools and AI models to protect their systems. We call this defensive architecture **GuardSwarm**, mirroring StuxSwarm's structure but redirecting each agent toward protection rather than exploitation. This mirrors how cybersecurity teams have historically worked, but with a more advanced toolkit. The fundamentals are still in place, despite the vastly superior attack vectors available today. In this section, a mirror to the agentic attack architecture describes a system with multiple agents assigned to specifically combat the main attack strategies outlined previously. Companies like Anthropic are already doing something similar, although less automated and focused on code, through Project Glasswing and its partners [13].

A. Human Prevention Layer

The first layer of prevention always has to be education on good practices of digital hygiene. This concept is already very common, and many companies require mandatory workshops that teach employees about common phishing strategies that attackers use [12]. With modern attack tools, employees must be much more careful and aware about what is out there and stay up to date with advanced social engineering strategies powered by AI. First, there must be a dedicated course on social media risks and demonstrations on how attackers can now use publicly available photos and videos to create convincing deepfake images, videos, and audio very quickly. This is not meant to restrict what a user can post regarding their personal life, but to inform them to be skeptical if they receive any concerning media or communications and to discern real threats from phishing attempts. Employees must also be taught to never comply with threats until they have been validated by law enforcement or the company's internal cybersecurity team. Even though the attacks may be very convincing, it is wise to confirm with other professionals before granting any requests. This lesson can be directly tied to the Arup case, where a simple confirmation before transferring money would have raised red flags and the attack would have failed [15]. In the case of a Stuxnet-style attack requiring physical USB access, it is best to disable any physical ports on sensitive systems to prevent third-party malware from being injected. Although the human layer is not the perfect defense against highly advanced attacks that exploit zero-day vulnerabilities, it can filter out common attacks such as financial fraud.

B. Defense Agent 1: Vulnerability Scanner

The first defense agent mirrors the exploit discovery agent in the attack architecture. The main task of the vulnerability

scanner agent is to beat the exploit discovery agent by finding vulnerabilities in the host system first. Claude Mythos or an equivalent model would power this agent and run 24/7 via cron scheduling to continuously find and patch any new bugs introduced [13]. The main output includes examples of exploits and proactively patching bugs as soon as they are found. This reduces the time to decide whether a bug is "important" enough to patch. Vulnerability scanner agents also ensure all systems are up to date on their assigned machine. Project Glasswing focuses mostly on finding bugs, and it is the companies' responsibility to implement patches (potentially also using Mythos), while this agent would have the combined responsibility [13]. Anthropic is offering $100M in credits for defensive scanning and patching for partners [13]. Some version of the vulnerability scanner agent has been adopted by partners including AWS, Apple, Broadcom, Cisco, CrowdStrike, Google, JPMorganChase, the Linux Foundation, Microsoft, NVIDIA, and Palo Alto Networks [13]. This gives confidence that such an agent has been validated at scale and has already been reported to have found thousands of zero-days in major operating systems and browsers [17].

C. Defense Agent 2: Social Media Monitor

Monitoring employee social media is something companies already do, and this agent would only automate the process. One thing to note is that the goal of this agent is not to monitor and report inappropriate content, but to protect employees from a cybersecurity perspective. This agent has the same capabilities as the reconnaissance agent and creates similar employee profiles with public images and videos, assigning a safety score based on what it found. This agent has an easier job than the reconnaissance agent since it already has the database of employee information and knows exactly which profiles to monitor. An additional task this agent completes is periodically monitoring for unusual traffic to certain profiles, suspicious emails and phone calls, and other probes that agentic attacks may use to reach an employee. If an employee is flagged as a potential target, this agent immediately notifies the employee and the main incident response conductor to correct the issue.

D. Defense Agent 3: Deepfake Protector

Cybersecurity should not interfere with employee personal lives for the most part. In cases where companies want to provide their employees with tools to protect themselves online without being too invasive, this agent can be requested to help. Deepfake protector agents have the ability to add adversarial perturbations to photos, videos, and audio before they are publicly uploaded to prevent attackers from being able to use them for malicious purposes [16]. AntiFake specifically makes voice cloning very difficult or impossible, reporting up to a 100% success rate on some configurations while being undetectable to the human ear [11]. Table IV summarizes the performance of AntiFake's two protection schemes.

If an employee receives a suspicious media sample they could report it to the deepfake protector agent, which would check for common signs of AI-generated media or built-in undetectable signatures such as Google's SynthID. This protector agent is available upon request from employees and will be made known via the human prevention layer course mentioned previously.

TABLE IV
ANTIFAKE PROTECTION PERFORMANCE ACROSS SPEAKER VERIFICATION SYSTEMS [11]. AEER = AUTO. EMBEDDING ERROR RATE. PSD = PERCEPTUAL SPEAKER DISSIMILARITY. MOS = MEAN OPINION SCORE (AUDIO QUALITY).

Scheme	**AEER**			**PSD**	**MOS**
	GMM	ivector	Azure		
Threshold	98.50%	98.75%	100%	4.85	3.44
Target	98.50%	99.50%	100%	4.88	3.32

E. Defense Agent 4: Malware Detector

Current antivirus software is effective at finding legacy-style malware, but there is an additional need for agentic systems that live permanently on device and scan for malware using LLM intelligence. The malware detector agent is used in conjunction with antivirus software, using common techniques that IT professionals would employ to diagnose a machine with a worm or any other malware [2]. Stuxnet was not detected by antivirus software automatically even though there were suspicious drivers installed on the system. A knowledgeable agent would have the ability to recognize that these drivers were not built into Windows or part of any known software, so they would be immediately flagged. Along with general filesystem browsing and scans with full root access, the agent has access to similarity hashing tools such as ssdeep and TLSH for known malware variants [2]. Polymorphic code has been a real challenge for antivirus software since an LLM could very quickly modify its own code, so the only real protection is using another LLM that can detect morphed worm variants much faster [10]. If a piece of software is flagged as malware, the agent has the ability to try and stop it in its tracks via filesystem access or temporary counter-worm software that is written and executed on the spot. Reports are sent to the incident response conductor, which handles the next steps.

F. Defense Agent 5: Incident Response Conductor

As with the agentic attack framework, the incident response conductor is the main agent that notifies the human (or cybersecurity team) responsible when an attack is detected or a vulnerability is found. This agent must run 24/7 and deploy the necessary subagents to deal with each task. In general, vulnerability scanners should be deployed on proprietary systems with sensitive information such as company servers that host databases or applications. A few social media monitors should be periodically deployed, depending on the size of the company, but it is not a requirement to have a dedicated agent for each employee. Deepfake protectors should be available on request when an employee receives a suspicious piece of media

or if they want to protect something before making it publicly available on social media. The malware detector agents should be the majority of the deployed agents and should be installed on any company device. The reason for this is to detect and contain any malware as soon as possible as an extension of antivirus software, which should already be installed on every company device. The exact agent count, agent coordination, and initial response will be handled by the incident response conductor, which escalates to human teams if necessary.

Figure 2 illustrates the GuardSwarm architecture, mirroring StuxSwarm from Figure 1.

G. The Arms Race

Both the attack and defense systems have access to the same software tools, but the main goals of the systems are configured slightly differently, and this leads to advantages on both sides. For the attacker side, these tools allow for spending less time on picking a specific target and instead specifying the parameters to look for and launching many parallel attacks, hoping some of them succeed. Attackers are able to filter out targets that have the correct protections in place and focus on easy targets with minimal security. Defenders know the modern infrastructure and attack strategies that attackers will use. This allows defenders to harden security ahead of time and be prepared when an attack is detected to prevent any damage. Companies must embrace a new system of agentic cybersecurity to keep up with new attacks and at least deter agentic attackers from attempting attacks. As AI models improve they will become more accessible to everyone, both attackers and defenders. More functionality is continually added to harnesses such as OpenClaw, which only extends the potential attack strategies that defenders must consider. This becomes a continuous improvement loop for attackers and defenders, leading to the never-ending cat-and-mouse situation that companies and individuals must adapt to.

VII. Discussion and Limitations

The systems described in the sections above are both theoretical, and there is no evidence that attackers are actively using OpenClaw or other agentic harnesses for real attacks. AI-enabled attacks thus far still require much human coordination, but this threat analysis shows that it is only a matter of time before attackers automate their techniques. It is important to have agentic defense systems in place before this happens to guarantee the first-mover advantage. Jailbreaking models is still not consistent, and model makers invest much time and research to prevent jailbreaking in the first place [4].

The main gap between a Stuxnet-style attack and full automation is still government-level intelligence and physical access to hardware devices to inject worms. General cyber-security hygiene has improved greatly as new generations of employees have grown up to be more tech-savvy, which is the main line of defense against potential attacks. Companies invest much more time and resources into protecting their systems, and this has only made it harder for cyber attackers with no access to physical hardware. As IoT devices become increasingly common, this also increases the number of lateral attack vectors that require no physical access, but there is no proof of a Stuxnet-level attack injected via IoT devices yet. There are experimental reports of Claude Code reverse-engineering IoT APIs completely autonomously and hacking devices connected to a smart home system. Attackers could exploit unsuspecting victims as technology gets integrated into common appliances in the office or at home, mirroring the printer attack capability of Stuxnet.

There are many ethical considerations when describing an agentic attack system, which is why this paper focuses on the theoretical threat analysis and not the fully detailed implementation, as well as describing the best defense against an agentic threat.

Coordinating between agents is still early, although it has been rapidly improving through harnesses like Claude Code and OpenClaw. Real-world implementation will still require advanced AI, IT, and cybersecurity knowledge at this point, but attackers could potentially sell malware-as-a-service packages to non-tech-savvy criminals. Deepfake quality generation from unrestricted models is still detectable by humans with a trained eye, and the state-of-the-art models are very heavily protected by makers such as OpenAI and Google. Anthropic has specifically decided to not release Claude Mythos at this time to give defense systems a head start, but if the trend of easier access to intelligence continues, this is only a temporary solution [17]. AI is still in its very early stages, and Anthropic's own system card warns that "rare instances of our models taking clearly disallowed actions" have been observed and, "in even rarer cases, seeming to deliberately obfuscate them" [17]. This means that even AI used defensively could behave unexpectedly. Giving agents full root access to a computer may lead to accidental disasters such as database deletion even if the model did not intend to do so.

Costs of running 24/7 agents have been reported to reach thousands of dollars per day, which may be inaccessible and impractical for individuals and small businesses who need the most protection. Companies must decide how protective their policy must be, as social media monitoring and employee privacy might be affected with instructions that are too restrictive. There are newer projects to introduce self-improving harnesses, which only add fuel to the fire of powerful agentic attack systems.

VIII. Conclusion

Stuxnet required nation-state-level resources and investment and took years of work by very specialized teams. This attack was specifically designed to target one nuclear enrichment facility, and it was almost flawless. Modern AI capabilities and tools can be mapped to every part of Stuxnet, and in theory nations with unlimited budgets could deploy automated attacks to many targets faster, cheaper, and more effectively. Common criminals can also take advantage of modern AI tools to target many lower-level victims for monetary gain. It is important to be aware of what such an agentic system could look like to properly design a defensive counterpart. StuxSwarm

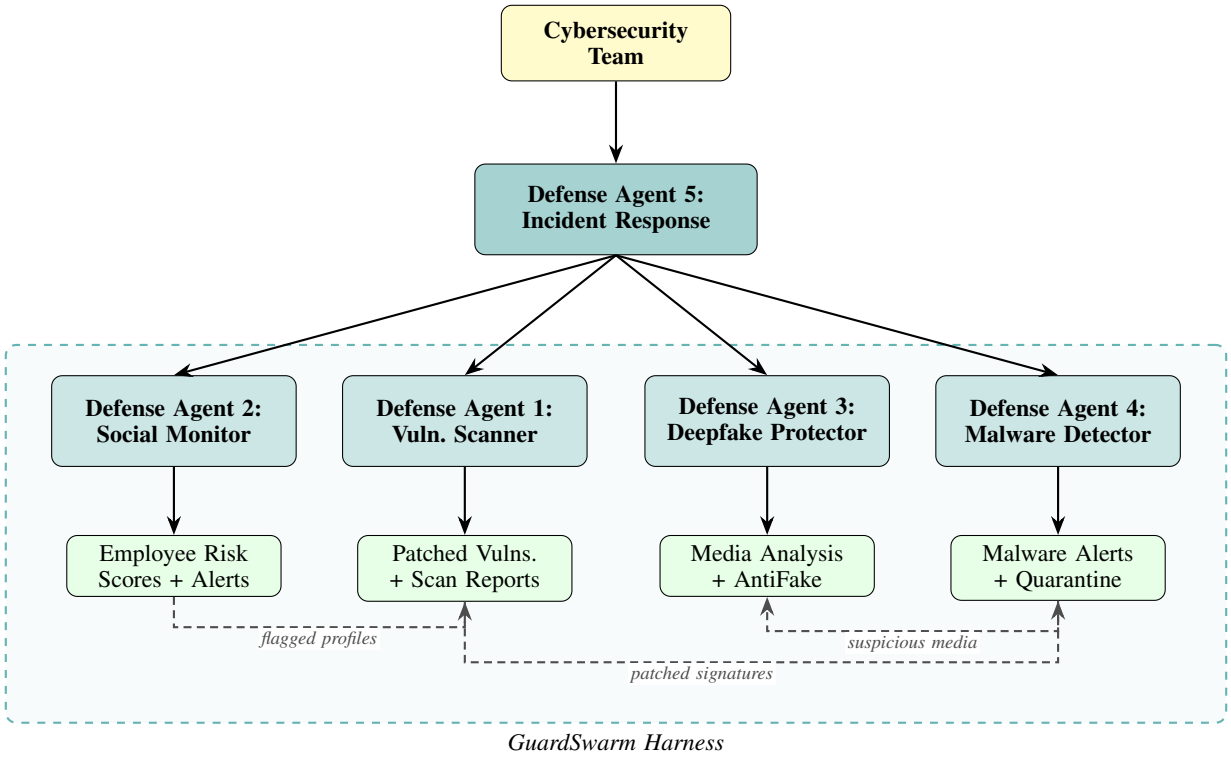

Fig. 2. GuardSwarm: proposed defensive architecture. The structure mirrors StuxSwarm from Fig. 1, with each defense agent countering a corresponding attack agent. Dashed arrows indicate shared intelligence between defense agents.

demonstrates the attack potential; GuardSwarm demonstrates how the same tools can be redirected toward protection. The age of agentic cybersecurity is here, and the AI arms race now demands proactive modern defense systems that organizations must deploy to defend against AI adversaries. Future work exploring both StuxSwarm and GuardSwarm would include real sandboxed implementations and performance comparisons for simulated scenarios that organizations could encounter in real life. This would reveal practical improvements necessary to make defense systems robust enough to be implemented in practice.

Acknowledgment

The author would like to thank Dr. Hamilton for guidance on this research topic. Claude Opus 4.6 (Anthropic, 2025) was used to assist with formatting this paper into IEEE LaTeX, inserting citation tags, and conducting literature searches to identify relevant references. All written content is the author's own work.

References

[1] H. Xu, S. Wang, N. Li, K. Wang, Y. Zhao, K. Chen, T. Yu, Y. Liu, and H. Wang, "Large language models for cyber security: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 2, pp. 1–39, 2025. doi: 10.1145/3769676.

[2] M. G. Gaber, M. Ahmed, and H. Janicke, "Malware detection with artificial intelligence: A systematic literature review," *ACM Comput. Surv.*, vol. 56, no. 6, art. 148, pp. 1–33, Jan. 2024. doi: 10.1145/3638552.

[3] F. He, T. Zhu, D. Ye, B. Liu, W. Zhou, and P. S. Yu, "The emerged security and privacy of LLM agent: A survey with case studies," *ACM Comput. Surv.*, vol. 58, no. 6, 2025. doi: 10.1145/3773080.

[4] Das *et al.*, "Security and privacy challenges of large language models: A survey," *arXiv preprint arXiv:2402.00888*, 2024.

[5] J. R. Lindsay, "Stuxnet and the limits of cyber warfare," *Security Studies*, vol. 22, no. 3, pp. 365–404, 2013. doi: 10.1080/09636412.2013.816122.

[6] P. W. Singer, "Stuxnet and its hidden lessons on the ethics of cyber-weapons," *Case W. Res. J. Int'l L.*, vol. 47, no. 1, pp. 79–95, 2015.

[7] A. Falco, "Stuxnet facts report," NATO Cooperative Cyber Defence Centre of Excellence (CCDCOE), Tallinn, Estonia, 2012. [Online]. Available: https://ccdcoe.org/uploads/2018/10/Falco2012_StuxnetFactsReport.pdf

[8] Google Threat Intelligence Group, "GTIG AI threat tracker: Advances in threat actor usage of AI tools," Google Cloud Blog, Nov. 2025. [Online]. Available: https://cloud.google.com/blog/topics/threat-intelligence/threat-actor-usage-of-ai-tools

[9] Check Point Research, "AI evasion: The next frontier of malware techniques," Check Point Blog, Jun. 2025. [Online]. Available: https://blog.checkpoint.com/artificial-intelligence/ai-evasion-the-next-frontier-of-malware-techniques/

[10] National Institute of Standards and Technology, "Adversarial machine learning: A taxonomy and terminology of attacks and mitigations," NIST AI 100-2e2025, Mar. 2025. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2025.pdf

[11] Z. Yu, S. Zhai, and N. Zhang, "AntiFake: Using adversarial audio to prevent unauthorized speech synthesis," in *Proc. 2023 ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, Copenhagen, Denmark, Nov. 2023, pp. 1–15. doi: 10.1145/3576915.3623209.

[12] T. T. Longtchi, R. M. Rodriguez, L. Al-Shawaf, A. Atyabi, and S. Xu, "Internet-based social engineering psychology, attacks, and defenses: A survey," *Proc. IEEE*, vol. 112, no. 3, pp. 210–246, Mar. 2024. doi: 10.1109/JPROC.2024.3379855.

[13] Anthropic, "Project Glasswing: Securing critical software for the AI era," Apr. 2026. [Online]. Available: https://www.anthropic.com/glasswing

[14] CrowdStrike, "2025 Global Threat Report," Feb. 2025. [Online]. Available: https://www.crowdstrike.com/en-us/global-threat-report/

[15] B. Chan, "Finance worker pays out $25 million after video call with deepfake 'chief financial officer,'" *CNN*, Feb. 4,

2024. [Online]. Available: https://www.cnn.com/2024/02/04/asia/deepfake-cfo-scam-hong-kong-intl-hnk

[16] H.-T. Nguyen-Le, V. Tran, T. Nguyen, and N.-A. Le-Khac, "A survey on proactive deepfake defense: Disruption and watermarking," *ACM Comput. Surv.*, vol. 58, no. 5, pp. 1–37, Nov. 2025. doi: 10.1145/3771296.

[17] Anthropic, "System card: Claude Mythos Preview," Apr. 2026. [Online]. Available: https://www-cdn.anthropic.com/8b8380204f74670be75e81c820ca8dda846ab289.pdf

[18] L. Vaas, "Scammers deepfake CEO's voice to talk underling into $243,000 transfer," Sophos Naked Security, Sep. 5, 2019. [Online]. Available: https://news.sophos.com/en-us/2019/09/05/scammers-deepfake-ceos-voice-to-talk-underling-into-243000-transfer/

[19] J. Huang, interview on *The Joe Rogan Experience*, episode #2422, Dec. 3, 2025. [Online]. Available: https://open.spotify.com/episode/0yT4ec9M6GobLC5ByN8pX3