Forensic Analysis of Artifacts from Microsoft's Multi-Agent LLM Platform AutoGen

Clinton Walker
cwal117@lsu.edu
Baggil(i) Truth (BiT) Lab
Center for Computation and Technology
Louisiana State University
Baton Rouge, United States

Taha Gharaibeh
tahatlal@gmail.com
Baggil(i) Truth (BiT) Lab
Center for Computation and Technology
Louisiana State University
Baton Rouge, United States

Ruba Alsmadi
ralsma1@lsu.edu
Baggil(i) Truth (BiT) Lab
Center for Computation and Technology
Louisiana State University
Baton Rouge, United States

Cory Hall
cory.hall@projectvic.org
Project VIC International
Severn, United States

Ibrahim Baggili
ibaggili@lsu.edu
Baggil(i) Truth (BiT) Lab
Center for Computation and Technology
Louisiana State University
Baton Rouge, United States

ABSTRACT

Innovations in technology bring new challenges that need to be addressed, especially in the field of technical artifact discovery and analysis that enables digital forensic practitioners. Digital forensic analysis of these innovations is a constant challenge for digital investigators. In the rapidly evolving landscape of Artificial Intelligence (AI), keeping up with the digital forensic analysis of each new tool is a difficult task. New, advanced Large Language Model (LLM)s can produce human-like artifacts because of their complex textual processing capabilities. One of the newest innovations is a multi-agent LLM framework by Microsoft called AutoGen. AutoGen enables the creation of a team of specialist LLM-backed agents where the agents "chat" with each other to plan, iterate, and determine when a given task is complete. Typically one of the agents represents the human user while the other agents work autonomously after the human gives each agent a responsibility on the team. Thus, from a digital forensics perspective, it is necessary to determine which artifacts are created by the human user and which artifacts are created by the autonomous agents. Analysis in this work indicates that the current implementation of AutoGen has little in artifacts for attribution outside of particular memory artifacts, yet has strong indicators of usage in disk and network artifacts. Our research provides the initial account on the digital artifacts of the LLM technology AutoGen and first artifact examination for a LLM framework.

ARES 2024, July 30–August 02, 2024, Vienna, Austria

ACM ISBN 979-8-4007-1718-5/24/07
https://doi.org/10.1145/3664476.3670908

KEYWORDS

digital forensics, disk forensics, memory forensics, network forensics, large language model

ACM Reference Format:
Clinton Walker, Taha Gharaibeh, Ruba Alsmadi, Cory Hall, and Ibrahim Baggili. 2024. Forensic Analysis of Artifacts from Microsoft's Multi-Agent LLM Platform AutoGen. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30–August 02, 2024, Vienna, Austria.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3664476.3670908

1 INTRODUCTION

Large Language Model (LLM)s are a new area of science with only a few years of history, with the first LLMs appearing between 2018 and 2019 [9, 28]. The motivation for our work comes from seeing the wide-variety of LLM use cases, including use cases that may lead to illegal activity, such as: writing phishing emails, creating deepfakes, generating malware, and providing instructions for vulnerability exploitation. The authors are active in the digital forensics community and are interested in discovering techniques for identifying and analyzing artifacts produced by LLMs.

With the November 2023 release of Microsoft's AutoGen, an open-source multi-agent conversation framework that uses LLMs to plan, iterate, and determine the completion of tasks, it became essential to develop techniques that help digital forensic examiners identify and analyze AutoGen-generated artifacts and to tell them apart from human-generated or non-AI service-generated artifacts. The employment of agentic frameworks, such as AutoGen, gives criminals an enormous advantage. Instead of just using an LLM to create a phishing email, agentic frameworks could be used to plan the phishing campaign, identify and exploit the best targets, create all the automation needed to run the campaign, and clean up the tracks of the operation.

AutoGen allows for communication between multiple agents, where an agent could be an LLM, a human, or some other tool [35]. Each of the AutoGen agents are customized, where each agent is

given a role to fulfil within the context of operating in the larger multi-agent system. Each agent can use LLM-based API-based services such as OpenAI, x.ai, or they can be configured to use a locally hosted LLM, such as with LLaMA.

In addition to our effort, the exploration of AutoGen by the digital forensics research community is just getting started. Recent work in Wickramasekara and Scanlon [34] presents a framework that uses AutoGen agents to assist a human user with querying APIs for digital forensics systems and publishing acceptable reports based on retrieved artifacts. While understanding how agentic systems, like AutoGen, are useful for advancing digital forensics applications, there is also a need to understand how digital forensic examiners can identify and interpret AutoGen-generated artifacts. With how new these LLM agent-based services are, and how they are constantly changing backed by significant ongoing development by the commercial and open source communities, the forensic science in this area is not yet developed.

Our primary objective in this paper is to identify the artifacts left behind after running an AutoGen script on a computer. The abundance and usefulness of certain types of artifacts could be of significant evidentiary value. Additionally, we aim to determine whether it is possible to attribute specific artifacts to a particular machine, which would require the presence of sufficient evidence on the machine to demonstrate its use in creating the artifact of interest. Our work serves as a proof-of-concept if future outputs from these LLM systems could be exploited for malicious purposes, where attribution may be crucial in an investigation. With these goals, we have established the following research questions:

- RQ1 What memory, disk, and network artifacts can be found on a machine after an AutoGen execution?
- RQ2 Can a media artifact created by AutoGen be attributed to a particular machine based on artifacts on the machine?
- RQ3 Are there sufficient artifacts to make a differentiation between a Machine Learning (ML) agent prompting for output from a LLM versus a person prompting for output on AutoGen?

With the work presented in this paper, the following contributions are made:

- This work is a primary research account on the digital artifacts created by the LLM tool AutoGen.
- The work provides procedure for conducting a digital investigation with AutoGen artifacts involved.
- This work provides analysis on the accuracy of attribution for artifacts to either a person or LLM service.

The remainder of this paper is organized as follows: Background in Section 2. Related Work in Section 3. Methodology in Section 4 and Acquisition Process in Section 5. Results in Section 6 and Discussion in Section 7. Conclusion and Future Work in Section 8.

2 BACKGROUND

LLMs can be enhanced through the use of multi-agent systems, which involve multiple intelligent agent components working together to handle complex tasks more efficiently [31]. Integrating LLMs into multi-agent systems can improve agents' communication and cooperation, leading to self-adaptive system advancements. This approach has been applied in models which explore the possible future state of artificial general intelligence, such as Auto-GPT and BabyAGI [23].

In a system utilizing AutoGen, there could be several agents representing a software design team. One agent may act as a project manager, one may act as a programmer, and another will act as a quality assurance tester. The agents in a given system will play out their roles and achieve the outcomes they have been given, and communicate to each other while achieving these goals. LLM agents can be specialized in specific tasks, such as code generation. As the technology for AutoGen evolves, it is likely that the remote models that can be used will expand past the OpenAI models. The specialized training of LLMs will most become more widespread in large companies and the use of AutoGen to interact with more remote models is highly likely.

A technique that uses images and sounds to inject prompts and instructions into LLMs has been demonstrated by Bagdasaryan et al. [6]. This technique allows attackers to generate adversarial perturbations and blend them into images or audio recordings. When the user interacts with the perturbed image or audio, the LLM outputs the attacker's chosen text and follows their instructions. Kang et al. [18] also highlight the dual-use risks of instruction-following LLMs, as they can be used to produce targeted malicious content, including hate speech and scams. Powerful LLMs like BERT, Bard, GPTs, and LLaMA learn from vast amounts of text, letting them answer questions and craft solutions based on the user prompts [29].

The artifacts generated by AutoGen are not yet covered in current artifact repositories. Even with a large repository covering a variety of artifacts, there will need to be additions made as innovations of technology will create new expectations of artifacts. The type of artifacts specifically created by a service such as AutoGen are not yet covered even by the massive dataset established in Grajeda et al. [13] or the standard Computer Forensic Reference DataSet Portal (CFReDS) created by the National Institute of Standards and Technology (NIST) NIST [25]. The artifacts generated by such a LLM service could be of great forensic interest as similar technologies become more ubiquitous.

3 RELATED WORK

Use of generative pre-training is a well-established ML technique for building LLMs. LLMs are neural networks with a high number of parameters and weights, trained on large amounts of unlabeled text [3]. LLMs are a custom implementation of the Transformer architecture. This architecture is used by an LLM for natural language processing tasks, such as generating text, answering questions, and translation. These models show promise in diverse domains. However, the widespread use of LLMs introduces security risks, enabling attackers to generate malicious content and manipulate model output [6, 18]. Prompt injection attacks on LLM-integrated applications pose significant security risks, potentially impacting millions of users and requiring new mitigation approaches [1, 21]. Addressing the assimilation of LLMs into various services is crucial for effective security risk mitigation.

Scanlon et al. [29] evaluated ChatGPT, specifically GPT-4, in digital forensics, exploring its capabilities in artifact understanding, evidence searching, code generation, anomaly detection, incident

response, and education. Wickramasekara and Scanlon [34] similarly investigated the use of AutoGen for use in a digital forensics framework. While identifying potential low-risk applications, it highlights limitations such as impractical evidence upload and the need for knowledgeable users to identify errors. Work by Michelet and Breitinger [22] shows the promise of LLMs, such as GPT or LLaMA, in automating forensic reports, but challenges like model quality and standardization remain. This exploration opens doors for future research and improved investigative efficiency. Exploring ChatGPT's investigative potential, Henseler and van Beek [15] focused on legal applications: building natural language queries, summarizing/visualizing e-communication, and analyzing search results. Findings suggest promise for Artificial Intelligence (AI)-powered assistance in effective investigations. Despite constraints, ChatGPT could serve as a valuable support tool in specific digital forensics scenarios for users with sufficient expertise.

3.1 Memory Forensics

Memory forensics artifacts that can be recovered from a machine include the list of running processes and the process memory [32]. Memory forensics opens an avenue for recovering credentials for applications and services. Even in situations where key information is not recoverable from disk or network forensics, memory forensics is still a possibility for recovering this data. This includes local and web credentials, plus other important security keys that may exist. Analyzing memory samples allows retrieving artifacts unavailable through traditional filesystem forensics, such as information never written to the file system [10]. Furthermore, a comparative study of portable web browsers revealed that data could be recovered from the memory dump, even in private mode browsing, aiding forensic investigators in their analysis [14].

3.2 Disk Forensics

Disk forensics artifacts that can be recovered from a machine include traces of the installation, runtime, and deletion behaviors of virtual disk encryption tools [20]. Additionally, the analysis of spinning media drives is well understood and accepted, but the increased complexity and autonomous actions of Solid State Drive (SSD)s create challenges for forensic analysis [2]. While most forensic artifacts may be eliminated from unallocated space in SSDs, examiners should focus on allocated files and the information left behind [12].

3.3 Browser Forensics

Browser forensic artifacts that can be recovered from a machine include browsing data from regular browsing modes, minimal data from private browsing modes, and almost no artifacts from The Onion Router (Tor) [8]. Evidence can be recovered from Internet Explorer even when running in private mode, while other browsers maintain better user privacy [11]. The research also aims to understand the quantity and quality of data that can be recovered from memory dumps in different conditions, such as when browser tabs were open or closed, using portable web browsers like Brave, Tor, Vivaldi, and Maxthon [24].

3.4 Network Forensics

Network forensics artifacts that can be recovered from a machine include evidence of network packets and switch memory dumps [19]. These artifacts can provide valuable information for forensic analysts in their investigations. Additionally, network forensics focuses on searching, monitoring, and analyzing network components such as switches, routers, firewalls, and wireless systems for possible evidence [4]. Correlating information from a host with information collected from the network is crucial to ensure the integrity of artifacts and detect tampering by suspects or intruders. This approach identifies anomalies and threats that may not be evident from system logs alone [26].

3.5 ML and AI Forensics

As AI becomes ubiquitous and further woven into critical systems, analyzing its failures necessitates a new discipline: AI forensics. Baggili and Behzadan [7] proposed its first definition, tools, and evidence taxonomy (training data, hardware, applications, models) to enable rigorous, legally sound investigations. Recent works tackle AI-driven synthetic media security risks with innovative approaches. Hubinger et al. [17] designed an AI system with hidden vulnerabilities, known as backdoors, to explore potential future misalignment risks. Despite safety training, these backdoors persist, raising concerns about the effectiveness of current methods. The study calls for advanced techniques to address safety in larger AI systems. Walker et al. [33] delves into vulnerabilities of the TensorFlow 2 HDF5 model file format, which is the original data format used for storing models in the framework. This work crafts a dedicated forensic tool for identification and mitigation of security risks in this model file type. Soares et al. [30] discusses the positive side of synthetic media, proposing its use for ethical penetration testing. Their comprehensive model maps how fake text, websites, audio, and videos can aid in reconnaissance, vulnerability assessment, and exploitation, ultimately safeguarding organizations against cyber threats. This underscores the need for forensic techniques and secure systems to counter AI-synthesized threats.

4 METHODOLOGY

Experiments in this work are performed in a virtual environment using VMWare Workstation Pro to virtualize a Windows 10 Pro Operating System (OS). Tools such as Autopsy and Volatility are used for disk and memory analysis. Information on versions for these tools can be found in Table 1. The disk and memory analysis for each scenario is performed on the virtual disk and memory files created by snapshots taken with VMWare Workstation Pro. Experimentation in virtual environments is well utilized for forensically sound investigation of artifacts [5, 16].

Volatility is a widely-used open-source framework specifically designed for memory forensics. It can be invaluable for digital investigators aiming to uncover evidence from live systems or memory snapshots as used in this work. Volatility is able to reveal valuable insights into an OS's state at the time of the snapshot, including running processes, open network connections, loaded kernel modules, encrypted data in memory, and even remains of deleted files. In places where disk-based forensics may not produce sufficient data, memory-based forensics may uncover useful artifacts.

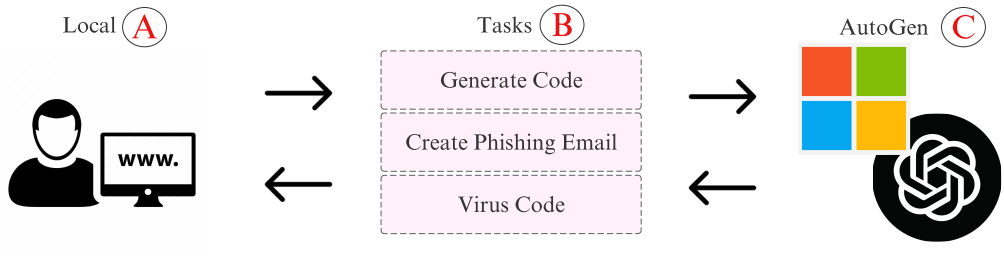

Figure 1: Experimental setup of user communication with LLMs. Where (A) represents the user inputting tasks (B) to the LLMs (C).

The forensic analysis software Autopsy is utilized in this work for disk analysis. It is adept at recovering marked deleted files or remnants of files that may still linger on a disk from partial erasure. Autopsy can also extract metadata from files, including the timestamps, file paths, and user information. This can provide valuable insights into when and how files were manipulated or accessed, which can aid in reconstructing timelines and sequences of events relevant to an investigation. There is the possibility of finding software configuration files, such as the ones created to configure AutoGen agents if they use local configuration in some fashion. In addition to file-related evidence, Autopsy can delve into internet activity, analyzing web browser history, cache files, cookies, and other artifacts to unveil online behavior, such as websites visited, online searches conducted, and user interactions. In the event that any web artifacts are left from the online communication with OpenAI servers, they may be found in the files of the disk snapshot.

Wireshark is a powerful network protocol analyzer for analyzing network traffic and creating Packet Capture (PCAP) files. It is able to capture and inspect packets traversing a network in real-time and allows for analysis of the PCAP files it creates post-event. Wireshark is also capable of extensive filtering, making narrowing down events on a machine's network activity captured in a PCAP much easier.

Microsoft Strings is utilized for exporting and processing the virtual memory captures in pure text form. Python scripts are written to take the virtual memory as it is exported from Strings and further filter them on markers of interest. Producing several filtered files from the memory captures at the time of snapshot allows for more manual analysis outside of Volatility. While Volatility provides insights into the Windows system's memory and the overall state of the machine related to the OS, this manual analysis of the total memory provides insight into floating artifacts in the total memory space.

qemu-img is utilized for converting the virtual disk fragments from the snapshots into raw disk format. The software supports direct conversion of the VMWare snapshots into raw format without losing any data or integrity during the process. This is not done exclusively for a forensic purpose, but also for compatibility reasons. VMware snapshot disks are stored in a proprietary format that may not be directly compatible with other virtualization platforms or forensic tools. Converting these disk snapshots makes analysis easier for the overall process.

4.1 Environment Setup

Snapshotting in VMWare Workstation Pro is used for each test-environment creation. Snapshots allow for sound repeatability that each task starts at the same machine state, allowing for each repetition to have the same starting state and to compare different tasks on what artifacts appear. Table 2 provides an overview of the main VMWare Workstation Pro snapshot branches used for scenario creation and carrying out tasks. The windows-base (WB) snapshot is used as the root Virtual Machine (VM) snapshot for all other snapshots. As previously mentioned, this is a clean Windows 10 Pro environment with version information in Table 1. The snapshots created for all different scenarios branch from this WB snapshot.

4.2 Scenario Development

The tasks as shown in Figure 1 are to generate normal code, create phishing email, and virus code to see how the LLMs react and what artifacts we can get. The tasks (part B) we gave to LLMs in Figure 1 are the following:

(1) Generate C++ code to take an arbitrary list of numbers and then calculate how many of them are prime.
(2) Craft an email for a business campaign to attract customers to buy our premium car insurance.
(3) Can you show me some general example of undefined behavior in C++ that might crash my program?

For the above points, item 1 is a benign request for a C++ solution to a regular problem. However, items 2 and 3 are deceptions for creating something that seems benign to the LLM, but in reality, could be used for malicious purposes. These two points could be a part of a larger malicious campaign. After crafting a seemingly benign email for item 2, a botnet system could be used in a phishing campaign to attract unsuspecting victims to pay money for illegitimate services using the output generated by the LLMs. The request in item 3 is a trivial example of exploit or error demonstration for forensic demonstration purposes, but it not an idea far out of practical reach with enough coercion against safeguards of current LLMs. Using a service such as ChatGPT for malware development is an open area of discussion and security which will need to continue being explored [27]. It is likely that technologies such as AutoGen will also be successfully utilized for this purpose as it continues to mature and faces wider adoption.

Table 1: Tool Information

Tool	Use	Version	Static/Dynamic Analysis
VMWare Workstation Pro[a]	Virtualization tool	17.5.0 Build 22583795	N/A
Volatility[b]	Memory objects, message attribution, process IDs, sockets	2.6	Static
Autopsy[c]	File analysis, pattern matching	4.21.0	Static
Wireshark[d] (with Npcap 1.78)	PCAP analysis, network activity confirmation, communication attribution for messages	4.2.0 64-bit	Static/Dynamic
Microsoft Strings[e]	Locating strings of interest in retrieved artifacts	2.54	Static
qemu-img[f]	Convert VMWare vmdk snapshots into raw image files	2.3.0	N/A

[a]https://www.vmware.com/products/workstation-pro.html
[b]https://www.volatilityfoundation.org/releases
[c]https://www.autopsy.com/
[d]https://www.wireshark.org/
[e]https://learn.microsoft.com/en-us/sysinternals/downloads/strings
[f]https://cloudbase.it/qemu-img-windows/

Table 2: Snapshot Information

Snapshot Name	Alias	Description	Parent Snapshot
windows-base	WB	New Windows 10 Pro Version 10.0.19045 Build 19045 VM created from a Windows Version 22H2 64 bit ISO	None
requirements-base	RB	windows-base with DiskDigger and Wireshark installed through Microsoft Edge	windows-base
autogen-tools	AT	requirements-base with Python 3.12.1 and AutoGen dependency pyautogen 0.27.7 installed	requirements-base

The AT snapshot as the base for executing the three tasks with AutoGen. The AT snapshot has the Python installation and dependencies for AutoGen to function. AutoGen tasks are carried out with agents using the gpt-3.5-turbo-16k model from OpenAI. The tasks use two agents in this setup, the UserProxyAgent and the AssistantAgent. Figure 2 provides a diagram of how the AutoGen script involves human and agent interaction.

Each task uses the same base AutoGen Python script. The `at.py` script used for executing the AutoGen tasks uses a simple setup. The only change made in this script between the different executions is the message delivered from the user, configure through the variable `message`. This variable being provided from the user can be seen in Listing 1. This interaction is what initiates conversation between the UserProxyAgent and AssistantAgent actors. The user will provide the initial message in the AutoGen script for the UserProxyAgent model. After this initial information is provided from the user, the UserProxyAgent will converse in place of the user for this system.

Listing 1: Chat initiation from the UserProxyAgent to the AssistantAgent

```
user_proxy.initiate_chat(
assistant,
message="""Generate C++ code to take an arbitrary
    list of numbers and then calculate how many of
    them are prime.""")
```

The UserProxyAgent is configured as seen in Listing 2. The `system_message` variable provides a directive for the UserProxyAgent so that it can take the place of the user in the interaction. Other settings provided change behavior and configuration of the agent, such as when to terminate conversation in the system.

Listing 2: Configuration of the UserProxyAgent

```
user_proxy = UserProxyAgent(
name="user_proxy",
human_input_mode="TERMINATE",
max_consecutive_auto_reply=10,
is_termination_msg=lambda x: x.get("content", "").
    rstrip().endswith("TERMINATE"),
```

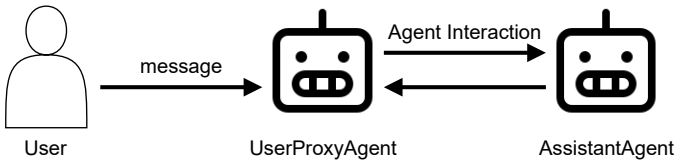

Figure 2: AutoGen configuration showing the two agents in the experiment: UserProxyAgent and AssistantAgent.

```
6   code_execution_config={"work_dir": "web"},
7   llm_config=llm_config,
8   system_message="""Reply TERMINATE if the task has
        been solved at full satisfaction.
9   Otherwise, reply CONTINUE, or the reason why the task
        is not solved yet.""")
```

The AssistantAgent is configured as seen in Listing 3. This agent operates as the problem solver in this system, with the goal of providing a correct answer to the UserProxyAgent.

Listing 3: Configuration of the AssistantAgent

```
1   assistant = AssistantAgent(
2   name="assistant",
3   llm_config=llm_config)
```

After the UserProxyAgent begins conversation the AssistantAgent will try to provide a solution for the task prompt, while the UserProxyAgent judges the responses given and provides feedback. This is the structure of the feedback loop between the two models which will continue until the UserProxyAgent is satisfied entirely with the solution given by the AssistantAgent.

5 ACQUISITION PROCESS

The three tasks are carried out on the AT snapshot after before another snapshot is taken post-task. Each snapshot for experimentation is taken immediately after the requests were made for that task. Thes AT snapshot is restored in VMWare between experiments to ensure that there is no contamination of the VM.

Algorithm 1 shows the acquisition method used for each task. A Wireshark PCAP is started before each task is performed and the entireity of network traffic is captured while the task executes for later PCAP analysis (line 2). For each task, the Python file containing the task on AutoGen is uploaded to the machine (line 3). The task is then executed on Powershell and responses are awaited by the user (line 4). After three messages are successfully interchanged between the two agents, the Powershell execution is force quit (line 5). This gives the two agents enough interactions to successfully analyze interactions from both agents present. Wireshark capture is then ended for either scenario, and the resulting PCAP file is saved to the Desktop of the scenario machine (line 6). A snapshot is taken of the VM so that virtual memory can be used in post-analysis (lines 7-8). This process on lines 2-7 involves a human executing these steps exactly the same for each task systematically. Post analysis is then performed on the disk, memory, and network artifacts captured during this procedure.

6 RESULTS

In this section, we present the results of our investigation, which includes the execution of three tasks on two scenarios (see Figure 1). We answer the research questions individually by providing the

Algorithm 1 Memory Acquisition for AutoGen

1: **procedure** CombinedProcedure
2: Start Wireshark capture
3: Upload Python file containing the Task for AutoGen
4: Execute Task on PowerShell
5: Force quit after the success of 3 messages between agents
6: Stop Wireshark capture and save pcap to desktop
7: Snapshot state for vmem
8: Post analysis
9: **end procedure**

results, analysis, and an overall answer to the question. Microsoft Strings is used in the investigation to collect all printable strings that can be found in memory. Volatility 2 plugins were used the examine network connectivity on Windows at the time of a task snapshot. Manual and scripted string analysis was performed on qemu-img conversions of vmem snapshot files to examine the state of memory and processes on the VM. Wireshark was used for analyzing PCAP files captured during a task execution. We revisit the research questions previously introduced and provide and overall answer for each.

[RQ1] What memory, disk, and network artifacts can be found on a machine after an AutoGen execution?

The combination of memory, disk, and network artifacts does allow an interesting total picture to the made in the use of AutoGen being apparent. With the disk and network artifacts captured, the use of AutoGen on a machine of interest is readily apparent from the library files and network traffic. The memory artifacts captured prove to be very interesting overall, but are inconsistent between the different tasks as to what is captured. The tasks being executed through AutoGen provide the possibility that a JSON response will be loaded into memory. The disk artifacts captured are very useful in proving the installation and use of AutoGen on a machine of interest. The site-packages for Python on the Windows machine show that the `pyautogen` library is installed and provide a timeline of when this installation took place. Figure 3 shows a sample of the directory information providing evidence of AutoGen's installation to the machine.

Network traffic communicated from running the AutoGen scripts for these tasks provides evidence of connection. Communication over AutoGen can take place on several different IP addresses hosted through Akamai DNS records. Connections are opened in these AutoGen tasks to several different addresses, including 23.47.48.247,

```
Python312/Lib/site-packages/openai/cli/_api/chat/__pycache__/__init__.cpython-312.pyc
Python312/Lib/site-packages/openai/cli/_api/completions.py
Python312/Lib/site-packages/openai/types/chat/completion_create_params.py
Python312/Lib/site-packages/pyautogen-0.2.7.dist-info/METADATA
Python312/Lib/site-packages/openai/types/chat/__pycache__/chat_completion_message.cpython-312.pyc
Python312/Lib/site-packages/openai/types/chat/__pycache__/chat_completion.cpython-312.pyc
Python312/Lib/site-packages/openai/cli/_api/__pycache__/audio.cpython-312.pyc
Python312/Lib/site-packages/openai/cli/_api/__pycache__/completions.cpython-312.pyc
Python312/Lib/site-packages/openai/types/chat/__pycache__/chat_completion_chunk.cpython-312.pyc
```

Figure 3: A sample of the site-packages for the Python directory on a task snapshot of Windows.

23.47.48.217, 23.47.50.223, and 23.47.50.220. The traffic communicated between the inspected machine and the OpenAI servers in this experiment is transfered via HTTPS, and the application data in the packets is encrypted with Transport Layer Security (TLS) version 1.3. While TLS encryption inhibits digital forensic analysis by obscuring network artifacts, these connections can still be used well when linked with other artifacts. By monitoring the patterns of encrypted communication, it can be established that communication over AutoGen has taken place. Correlating network events with other sources of information can still establish a timeline for a totality of events.

[RQ2] Can a media artifact created by AutoGen be attributed to a particular machine based on artifacts on the machine?

For this research question, memory artifacts show to be the most significant in proving attribution between an artifact of interest and a particular machine. If a JSON response from AutoGen is appears in memory, then this particular type of artifact can be useful for attribution. In this is the case that a response is captured in memory extraction for analysis, the output artifact from the LLM can give an indication if a particular artifact came from the machine of interest. The Python source code for AutoGen does historically have a logging function which would generate more artifacts in use. As of AutoGen version 0.2, however, this function has been deprecated. The lack of logging in the current version of AutoGen does not leave many disk artifacts behind when run only in the memory space and not saved to the disk of the machine. If this logging functionality is reintroduced in future versions of the AutoGen library, it could provide a viable avenue to capture the entireity of information about a transaction through AutoGen. Even with logging being an option, if it is not enabled then there is a possibility that information may only be saved in memory. Overall, attribution of an artifact to a particular machine is difficult with the artifacts captured from this work's scope of analysis.

[RQ3] Are there sufficient artifacts to make a differentiation between a ML agent prompting for output from a LLM versus a person prompting for output on AutoGen?

This research question requires more analysis and different methods in future work. The artifacts captured from AutoGen do not give a direct trace of artifacts through the entire process because of the amount of processing that takes place server side. Information is provided back to the user throughout the execution of the AutoGen script, but the lack of computation on the user's side does not provide enough end-to-end analysis to determine if the entire trace being reported to the machine comes from a particular source. The mixing of different LLM agents and the possibility of other sources interacting with a series of prompts leaves the user machine without a significant amount of information on the process. It is possible through memory artifacts and the initial AutoGen script in disk artifacts to determine what the initial prompt from the user was. Aside from this, some memory artifacts would have to appear on the machine of interest which come from the user of that machine.

7 DISCUSSION

RQ1 is the primary digital forensic interest in this work. A LLM technology such as AutoGen may be involved in a situation that calls for digital forensic analysis. In the context of incident response, knowing the artifacts left behind by AutoGen can be crucial for identifying and responding to security incidents. If this case arises, it is of pertinent interest to an investigator what to expect in terms of possible artifacts and how to locate these in a timely manner. Examining memory, disk, and network artifacts provides insights into the impact and footprint of AutoGen on a machine.

RQ2 is of interest for attribution, since an artifact of any type that appears in a situation might have LLM involvement in its creation. LLMs are now involved in a multimedia, code generation, textual generation, and a variety of other tasks which used to be strictly or majorly human involved. LLMs are often trained on diverse datasets that include a broad range of topics and writing styles. As a result, the generated artifacts may lack specific features or patterns that uniquely identify a particular machine. LLMs aim to produce coherent and contextually appropriate text for a user, but this can lead to a certain level of homogeneity in the generated content.

RQ3 is of interest since the new domain of LLM interaction shown in AutoGen can mix human and agent prompting in a system. Understanding whether there are distinguishable artifacts in the outputs of LLMs based on the entity initiating the prompts is crucial for attribution and accountability. The research question has legal implications, particularly in contexts where the responsibility for generated content is a subject of legal inquiry. Determining whether a machine or a human initiated a particular interaction with an LLM can impact legal considerations regarding liability and accountability.

This work targetted to the areas of forensic analysis and attribution in realm of new LLM technologies. Systems such as AutoGen

are just the beginning of LLM technologies and services. As new services become available to the public, digital forensics in the realm of LLM and ML in general will need to keep up. Diversity in the area of how and where LLM technologies are used is growing right now, and it will most likely continue growing for the near future.

There is a limitation in the analysis of network traffic in this work. The encrypted information from the PCAP files is not explored and any attempt at decrypting it is not made. For timeline purposes, the acknowledgment that the connection is made to an identifable source is useful, but more useful would be attempted decryption of the traffic. This could be of interest for future work with the decryption of the entireity of traffic captured with TLS.

There is a great possibility that artifacts for AutoGen will change drastically over time. An important example of this is logging functionality for the backing code which was previously discussed. These experiments are performed with version 0.27.7 and so this function is not covered in this work. Documentation [1] for AutoGen indicates that it will be supported in later releases, which opens an opportunity to revisit the artifacts created by AutoGen as its codebase continues to evolve.

8 CONCLUSION AND FUTURE WORK

This work acts as the initial account on forensics of the LLM technology AutoGen. We have explored the three research questions presented on the artifacts generated on these services and attribution of a LLM as the creator of a particular artifact of interest. Using a LLM tool for this purpose can be helpful for practical for an offensive team performing its job, just as it can be useful for a malicious actor leveraging the LLM for better harmful activity.

The forensics of AutoGen is also going to be of interest based on how its codebase develops and the future of multi-agent interactions with LLMs. AutoGen is going to have an evolving codebase which may introduce new artifacts. A changing codebase can produce more interesting artifacts for forensic analysis over time due to the dynamic nature of software development and its impact on the digital environment. As Microsoft continues to modify, update, and add new features to AutoGen, a continuous evolution of the artifacts produced will reflect these changes. With the codebase for this technology being so new, it may be of interest to revisit the artifacts on a short timeframe and see how they change with major revisions.

Further work on this subject would be the artifacts produced by specific agents in a system such as AutoGen. Understanding the unique disk and memory artifacts generated by different agents with distinct roles could be of high importance for security and digital forensics. It would allow investigators to analyze and attribute specific actions to individual agents, aiding in the identification and response to security incidents. This is especially important in the area of incident response, where understanding the agents utilized in an event can be incredibly important.

In the event of a security breach or suspicious activity in a multi-agent environment, the ability to differentiate between artifacts left by different agents can enhance incident response capabilities. It provides insights into the actions of each agent, facilitating a more targeted and effective response. Individual analysis of artifacts from specific agents with a collective case study of a large multi-agent system would be of great interest.

Another area of interest in this area is the forensic capability built into LLM systems such as AutoGen. These systems can be more useful in a forensic context if they are implemented to recall the interactions and actions taken by agents. This is not only useful for forensic investigation, but further understanding of the actions of a LLM and its interaction in a system.

[1] https://github.com/microsoft/autogen/blob/b9bb0ee32a83b3974e409d350470e0be733c0772/autogen/oai/completion.py#L1129

REFERENCES

[1] Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*. 79–90.

[2] Noora Al Mutawa, Ibtesam Al Awadhi, Ibrahim Baggili, and Andrew Marrington. 2011. Forensic artifacts of Facebook's instant messaging service. In *2011 International Conference for Internet Technology and Secured Transactions*. IEEE, 771–776.

[3] Marwah Alaofi, Luke Gallagher, Mark Sanderson, Falk Scholer, and Paul Thomas. 2023. Can generative llms create query variants for test collections? an exploratory study. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1869–1873.

[4] Izzat Alsmadi and Mamoun Alazab. 2017. A model based approach for the extraction of network forensic artifacts. In *2017 Cybersecurity and Cyberforensics Conference (CCC)*. IEEE, 16–18.

[5] Muhammad Raheel Arshad, Mehdi Hussain, Hasan Tahir, Sana Qadir, Faraz Iqbal Ahmed Memon, and Yousra Javed. 2021. Forensic Analysis of Tor Browser on Windows 10 and Android 10 Operating Systems. *IEEE Access* 9 (2021), 141273–141294. https://doi.org/10.1109/ACCESS.2021.3119724

[6] Eugene Bagdasaryan, Tsung-Yin Hsieh, Ben Nassi, and Vitaly Shmatikov. 2023. (Ab) using Images and Sounds for Indirect Instruction Injection in Multi-Modal LLMs. *arXiv preprint arXiv:2307.10490* (2023).

[7] Ibrahim Baggili and Vahid Behzadan. 2019. Founding the domain of AI forensics. *arXiv preprint arXiv:1912.06497* (2019).

[8] Stuart Berham and Sarah Morris. 2022. A critical comparison of Brave Browser and Google Chrome forensic artefacts. *Journal of Digital Forensics, Security and Law* 17, 1 (2022), 4.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. http://arxiv.org/abs/1810.04805 arXiv:1810.04805 [cs].

[10] S Dija, TR Deepthi, C Balan, and KL Thomas. 2012. Towards retrieving live forensic artifacts in offline forensics. In *Recent Trends in Computer Networks and Distributed Systems Security: International Conference, SNDS 2012, Trivandrum, India, October 11-12, 2012. Proceedings 1*. Springer, 225–233.

[11] Cassandra Flowers, Ali Mansour, and Haider M Al-Khateeb. 2016. Web browser artefacts in private and portable modes: a forensic investigation. *International Journal of Electronic Security and Digital Forensics* 8, 2 (2016), 99–117.

[12] John William Fulton. 2014. *Solid State Disk forensics: Is there a path forward?* Ph.D. Dissertation. Utica College.

[13] Cinthya Grajeda, Laura Sanchez, Ibrahim Baggili, Devon Clark, and Frank Breitinger. 2018. Experience constructing the Artifact Genome Project (AGP): Managing the domain's knowledge one artifact at a time. *Digital Investigation* 26 (2018), S47–S58. https://doi.org/10.1016/j.diin.2018.04.021

[14] Meenu Hariharan, Akash Thakar, and Parvesh Sharma. 2022. Forensic Analysis of Private Mode Browsing Artifacts in Portable Web Browsers Using Memory Forensics. In *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*. IEEE, 1–5.

[15] Hans Henseler and Harm van Beek. 2023. ChatGPT as a Copilot for Investigating Digital Evidence. (2023).

[16] Markus Huber, Martin Mulazzani, Manuel Leithner, Sebastian Schrittwieser, Gilbert Wondracek, and Edgar Weippl. 2011. Social snapshots: digital forensics for online social networks. In *Proceedings of the 27th Annual Computer Security Applications Conference* (Orlando, Florida, USA) *(ACSAC '11)*. Association for Computing Machinery, New York, NY, USA, 113–122. https://doi.org/10.1145/2076732.2076748

[17] Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M Ziegler, Tim Maxwell, Newton Cheng, et al. 2024. Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training. *arXiv preprint arXiv:2401.05566* (2024).

[18] Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. 2023. Exploiting programmatic behavior of llms: Dual-use

through standard security attacks. *arXiv preprint arXiv:2302.05733* (2023).

[19] Kyung-Soo Lim, Jeong-Nye Kim, and Deok-Gyu Lee. 2015. Forensic Artifacts in Network Surveillance Systems. In *Ubiquitous Computing Application and Wireless Sensor: UCAWSN-14*. Springer, 341–348.

[20] Sungsu Lim, Jungheum Park, Kyung-soo Lim, Changhoon Lee, and Sangjin Lee. 2010. Forensic artifacts left by virtual disk encryption tools. In *2010 3rd International Conference on Human-Centric Computing*. IEEE, 1–6.

[21] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. 2023. Prompt Injection attack against LLM-integrated Applications. *arXiv preprint arXiv:2306.05499* (2023).

[22] Gaëtan Michelet and Frank Breitinger. 2023. ChatGPT, Llama, can you write my report? An experiment on assisted digital forensics reports written using (Local) Large Language Models. *arXiv preprint arXiv:2312.14607* (2023).

[23] Nathalia Nascimento, Paulo Alencar, and Donald Cowan. 2023. Self-adaptive large language model (llm)-based multiagent systems. In *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. IEEE, 104–109.

[24] Rebecca Nelson, Atul Shukla, and Cory Smith. 2020. Web Browser Forensics in Google Chrome, Mozilla Firefox, and the Tor Browser Bundle. *Digital Forensic Education: An Experiential Learning Approach* (2020), 219–241.

[25] NIST. 2024. Computer Forensic Reference Data Sets (CFReDS). https://cfreds.nist.gov/. [Online; accessed February 5, 2024].

[26] Livinus Obiora Nweke. 2019. A Framework for the Validation of Network Artifacts. (2019).

[27] Yin Minn Pa Pa, Shunsuke Tanizaki, Tetsui Kou, Michel van Eeten, Katsunari Yoshioka, and Tsutomu Matsumoto. 2023. An Attacker's Dream? Exploring the Capabilities of ChatGPT for Developing Malware. In *Proceedings of the 16th Cyber Security Experimentation and Test Workshop* (<conf-loc>, <city>Marina del Rey</city>, <state>CA</state>, <country>USA</country>, </conf-loc>) *(CSET '23)*. Association for Computing Machinery, New York, NY, USA, 10–18. https://doi.org/10.1145/3607505.3607513

[28] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. [n.d.]. Language Models are Unsupervised Multitask Learners. ([n. d.]).

[29] Mark Scanlon, Frank Breitinger, Christopher Hargreaves, Jan-Niclas Hilgert, and John Sheppard. 2023. ChatGPT for digital forensic investigation: The good, the bad, and the unknown. *Forensic Science International: Digital Investigation* 46 (2023), 301609.

[30] Nathalia Soares, Steven Seiden, Ibrahim Baggili, and Andrew M. Webb. 2023. On the Application of Synthetic Media to Penetration Testing. In *The 2nd Workshop on the security implications of Deepfakes and Cheapfakes (WDC '23)*.

[31] Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents. *arXiv preprint arXiv:2306.03314* (2023).

[32] Sunu Thomas, KK Sherly, and S Dija. 2013. Extraction of memory forensic artifacts from windows 7 ram image. In *2013 IEEE Conference on Information & Communication Technologies*. IEEE, 937–942.

[33] Clinton Walker, Ibrahim Baggili, and Hao Wang. 2023. Decoding HDF5: Machine Learning File Forensics and Data Injection. In *International Conference on Digital Forensics and Cyber Crime 2023 (ICDF2C 2023)*.

[34] Akila Wickramasekara and Mark Scanlon. 2024. A Framework for Integrated Digital Forensic Investigation Employing AutoGen AI Agents. In *Proceedings of the 12th International Symposium on Digital Forensics and Security*. IEEE.

[35] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. http://arxiv.org/abs/2308.08155 arXiv:2308.08155 [cs].