DAWOOD UNIVERSITY OF ENGINEERING AND TECHNOLOGY
GULBERG TOWN KARACHI-74800 (PAKISTAN)

FACULTY OF INFORMATION & COMPUTING SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Title: FILELESS MALWARE DETECTION AND ITS MITIGATION IN OPERATIONAL TECHNOLOGY

Submitted by:

Farayha Hydari 21F-BSCY-04
Syed Farman Hussain 21F-BSCY-26
Faheem Hussain 21F-BSCY-42
Usman Ahmed 21F-BSCY-49

Supervised by:

Dr. Ahmed Sikandar

DAWOOD UNIVERSITY OF ENGINEERING AND TECHNOLOGY
GULBERG TOWN KARACHI-74800 (PAKISTAN)

<u>FACULTY OF INFORMATION & COMPUTING SCIENCES</u>
<u>DEPARTMENT OF COMPUTER SCIENCE</u>

CERTIFICATE

This is to certify that the work present in this FYP thesis "fileless malware detection and its mitigation in operational technology" has been carried out by group members under the supervision of Dr. Ahmed Sikandar. The work is genuine, original and in our opinion it fulfills the requirements for FYP-II

Farayha Hydari 21F-BSCY-04

Syed Farman Hussain 21F-BSCY-26

Faheem Hussain 21F-BSCY-42

Usman Ahmed 21F-BSCY-49

Supervisor:

____

Co-Supervisor:

____

FYP Coordinator	Chairperson
____________________	____________________________
Dr. Abdullah Lakhan	Dr. Asif Aziz
Assistant Professor	Associate Professor

DAWOOD UNIVERSITY OF ENGINEERING AND TECHNOLOGY
GULBERG TOWN KARACHI-74800 (PAKISTAN)

FACULTY OF INFORMATION & COMPUTING SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

DECLARATION

We, the undersigned, declare that this thesis, titled Fileless Malware Detection and Its Mitigation in OT Systems, submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering at Dawood University of Engineering and Technology, is our original work, carried out under the supervision of our project advisor. The work presented herein has not been submitted, in whole or in part, for any other degree or qualification at this or any other academic institution.
We further declare that, to the best of our knowledge, the content of this thesis does not infringe upon any copyright or intellectual property rights. All sources, including published and unpublished works, have been duly acknowledged and referenced in accordance with academic standards. Contributions from team members, including the design, implementation, and evaluation of the proposed framework, were collaborative and equitable. Any external assistance, such as guidance from our advisor or use of open-source tools, has been appropriately recognized in the thesis.
We accept full responsibility for any errors or omissions in this work and confirm that the research was conducted with integrity and adherence to ethical principles.

ACKNOWLEDGEMENT

We extend our sincere gratitude to several individuals and entities who contributed to the successful completion of this Final Year Project, "Fileless malware detection and its mitigation in Operational Technology."

First and foremost, we express our profound appreciation to our supervisor, Dr. Ahmed Sikandar, for his invaluable guidance, insightful feedback, and continuous support throughout every stage of this research. His expertise and encouragement were crucial in shaping this project from conception to completion.

We are also thankful to Dawood University of Engineering and Technology, the Faculty of Information & Computing Sciences, and the Department of Computer Science for providing the necessary resources and a conducive environment for this research.

DAWOOD UNIVERSITY OF ENGINEERING AND TECHNOLOGY
GULBERG TOWN KARACHI-74800 (PAKISTAN)

FACULTY OF INFORMATION & COMPUTING SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

ABSTRACT

Recognizing that Operational Technology (OT) systems play an essential role in managing critical infrastructure such as power grids and manufacturing operations, we assessed the serious threat posed by fileless malware that lives in memory and bypasses antivirus agents. Fileless malware is particularly effective in OT environments where native deterrent technologies may not only be limited, but continuous operation is often required. The investigation outlined in this thesis suggests a modular approach to detect and remediate fileless malware in OT environments, but required we first evaluate common challenges such as legacy hardware and limitations in resources. The modular framework involved process monitoring through the Python psutil library to assess CPU load and highlight the high-CPU processes, in addition to what we call "memory forensics" through the use of WinPMem and Volatility to identify in-memory malware attacks such as process injections. Lastly, we used Suricata (an open-source network security monitoring software) with custom rules tailored to detect command and control activity utilizing malicious PowerShell scripts. Critical to the operational success, we implemented a rapid alerting mechanism with simple email notifications via smtplib and notifications through plyer to alert system administrators. The entire solution was built around a Flask-based web interface to present users with a straightforward dashboard to monitor active threats and respond to incidents. We tested the modular framework in a simulated Windows-based OT environment intended to illustrate characteristics of real world industrial control systems (ICS), and fed it several scenarios leveraging PowerShell scripts, WMI-based exploitation and process injections. We were able to develop a working modular framework where fileless malware was identified and remediated 70-95% of the time, which included a trustworthy false positive rate between 0-10%, while utilizing very low resource utilization (6% CPU and 115 MB of memory). The overall configuration allowed for additional detection protection with low overhead for resource-constrained OT environments. When the underlying research and findings are combined with the relatively lightweight open-source framework presented in the paper, there is a potential to increase the security of OT systems, particularly due to fileless malware's unique properties in FT environments. There were a couple of limitations to note, including our reliance on simulated testing and some false positives due to constant static thresholds. Future research work might consider the incorporation of dynamic thresholds and machine learning techniques, as well as a final validation within a real-world OT environment. Overall, this research contributes valuable research towards the reliability and security related to critical infrastructure management practices and align with similar standards outlined in IEC 62443 and NIST SP 800-82.

Keywords: Fileless Malware, Operational Technology, Cybersecurity, Memory Forensics, Suricata, Process Monitoring

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
OT	Operational Technology
ICS	Industrial Control Systems
SCADA	Supervisory Control and Data Acquisition
PLC	Programmable Logic Controller
HMI	Human-Machine Interface
EDR	Endpoint Detection and Response
IOC	Indicator of Compromise
C2	Command and Control
IDS	Intrusion Detection System
RAM	Random Access Memory
CPU	Central Processing Unit
WMI	Windows Management Instrumentation
HTTP	Hypertext Transfer Protocol
SID	Signature ID (in Suricata Rules)

CHAPTER 1

INTRODUCTION

1.1 Background

The operations of our key services depend on Operational Technology (OT). which consist of industrial control systems (ICS) as well as supervisory control and data acquisition. Another type is (SCADA) systems, also distributed control systems (DCS), and programmable logic automotive manufacturers install controllers (called PLCs) and use human-machine interfaces (called HMIs). They take care of managing important services like power grids, plants that clean water, and factories. Both transportation networks and systems are looked after to guarantee high performance and public safety. IT systems main focus are confidentiality and integrity, but MIS systems also consider availability. when discussing availability, OT systems give importance to safety, availability, and integrity since they are in charge of directly controlling physical events. Due to this distinction, occupational therapy happens in these settings. fileless malware is one of their biggest cyber threats. causes a major problem since it can hide in RAM and sneakily spreads through the computer.

Fileless malware is a sophisticated danger that works only in the background. focusing on system memory instead of creating disk-based artifacts typical to old Detection of malware is possible with antivirus technology. It is important to make use of system tools that are designed for security. for example, PowerShell, Windows Management Instrumentation (WMI), and scripts macros make fileless malware capable of mixing with common system tasks to stay hidden. Signature analysis is carried out with detection methods. In situations where legacy systems are found often, organizations experience a shortage of computing power and old software when looking for malicious activities. Handling these issues is much more difficult than dealing with other problems. The meeting of Information Technology with Industry 4.0 projects have resulted in more difficulties for OT networks. by giving attackers more opportunities to attack the IT systems to get access to and compromise OT [?]. Famous events, including the attack with Stuxnet on Iran's nuclear centers, The Industroyer attack on Ukraine's power grid in 2016 and what happened in 2010, prove that these incidents were highly destructive. threats from malware to OT environments are possible. Even though Stuxnet was not only This attack didn't make use of stored files, showing how attacks in memory might damage the system. Similarly, Industroyer carried out its attacks by modifying industrial control messages through fileless techniques and caused many power outages to take place. We can see from these situations why OT systems require special ways of discovering and handling issues caused by their limited resources, constant working role, and the need to remain compatible with old systems. The proposal will overcome these challenges by making a framework that helps detect and stop fileless malware in OT systems. Monitoring user processes, examining memory for attackers, Intrusion Detection Systems based on Suricata, and real-time emails provide a full security solution. Combining all these techniques, the system is capable of finding signs of cyberattacks quickly in OT while keeping operations running as smoothly as possible.

1.2 Problem Statement

Fileless malware is very dangerous for OT systems as it manages to bypass standard defenses security measures. This malware is different from conventional malware since its attack is not easy to spot. Rather than leave traces on the disk, fileless malware runs completely in memory by using standard programs. Many attackers use PowerShell and WMI to steal data from the system. Privilege gain, interference with the system's processing, and planned attacks [?]. Work environments in OT settings- The top priorities are that the system is available and safe, as such attacks may lead to disruption in the operations. The threats are related to stops in production, breakage of machinery, and potential dangers to people. An example of this is Triton malware attacked safety instrumented systems at an oil factory in Saudi Arabia. Plants revealed that fileless malware has the ability to harm critical safety controls [?]. The traditional forms of antivirus software are not very effective these days. It is a main challenge because there are no files to help with detection. Behavioral Even though analysis tools are promising, they sometimes issue wrong alerts in OT frameworks.

Sometimes, the right process can show unusual behavior if it is required to handle heavy demands. Also, OT systems deal with special challenges, one of which is hardware that was built years ago. Lack of computing resources, fewer times for upgrades, and constant maintenance criterion difficulties that affect smooth operations cause trouble with introducing expensive security resources. Because of the link between IT and OT networks, the risks are now higher. New ways for attackers to move from IT to OT mean that OT systems are under greater risk [?]. OT areas are further at risk since no specific ways have been made to spot fileless malware these challenges. Most of these tools are created for IT environments and do not work well in others for real time operation and minimal resources needed in OT, special precautions are taken downtime. Also, application whitelisting and fewer beliefs on what data is safe keep data safe in the cloud. Many OT systems do not integrate architecture because of compatibility problems by using old software. This project fills these gaps by designing a solution that fits your company system that uses easy-to-detect approaches and efficient ways to stop attacks plans, making sure an OT environment is both safe and works properly.

1.3 Objectives

This project seeks to come up with, execute, and evaluate a modular design methodology. A system for spotting and stopping fileless malware in OT environments. The specific objectives are explained below:

1. To use psutil in a process monitoring module to spot suspicious activities functions that need a lot of CPU and memory help in finding issues early what fileless malware can do during its operation.
2. To use procdump and Volatility tools to implement a memory forensics module Download and look at memory dumps to find out about possible memory-related malicious activities For example, these methods include PowerShell script execution and process injection.
3. Integrating Suricata, an open-source IDS system, is done in this way. Updated rules are created to look for fileless malware patterns in the system. PowerShell commands and attack on WMI, using network traffic.
4. How to develop an email and desktop notification process using smtplib and plyer report threats to administrators when detected, so they can respond straight away mitigation actions.
5. To make a web interface with Flask and Bootstrap to enhance ease of use system where I can watch my processes, take memory dumps, and use Suricata It allows administrators to be informed and handle new risks efficiently.
6. To check how well the framework operates in a digital environment, making sure the model is precise at spotting cases, rarely mistakes a healthy patient, and has a small effect on the system system performance.
7. Their purpose is to handle the uncommon problems caused by fileless malware in OT systems using a number of detection tools and taking steps in advance to combat issues supporting software that works even in environments where resources are not abundant.

1.4 Scope

The focus of the project is on building a way to find and manage fileless attacks. Malware threats in Windows-based OT settings are a typical occurrence in industrial companies control systems. It relies on available open-source tools such as psutil. To analyze the running processes, use procdump or Volatility, and for memory analysis use Suricata If you want to use network-focused intrusion detection, use peace and smtplib/plyer for fast alerts. The Flask is used to create the system as a web application that centralizes resources. System for tracking and controlling threats to the organization. These are some of the aspects covered within the scope:

- **Environment:** Windows 10 simulator that shows the characteristics of common ICS environments they are employed along with SCADA systems and PLCs.
- **Methods to Detect Threats:** Keep an eye on CPU, memory, and other apps identifying IOCs within volatile memory with memory forensics and with Suricata-based analysis, using custom-made network rules to find fileless malware. Use taskkill to end the process or set alerts for use on your desktop or email straightforward response by the organization, and a web tool for monitoring potential threats.
- Limitations: Because its design relies on a set of limitations, Legacy hardware and limited computational resources are included in OT systems and it must have as little

downtime as possible. It is not implemented in real OT systems because of the need for safety and limitations involved in operations, researchers simulate the system to gain results.

Both machine learning algorithms and zero-trust architecture have been mentioned in the research paper, which are not within the current implementation plans, are still in future work.

1.5 Scope of the Project

Fileless malware that evolves and grows more sophisticated is a big danger to OT systems. A basis for vital services that include energy, water supplies, and manufacturing. If such systems are attacked successfully, it may cause big economic and safety issues. Hazards, and problems with essential services, were demonstrated by earlier events. Stuxnet and Triton are examples of cyber-attacks. The framework provides strategies to handle this challenge. A customized system meant to combine several methods of detection— using the tasks of process monitoring, memory analysis, and a Suricata-based IDS as main tools in one method designed to meet the needs of occupational therapy. This project provides benefits for OT cybersecurity.

1.6 Organization of the Thesis

This thesis takes the following structure:

Chapter 2: Background and Literature Review covers a summary of the background of the study. Learning about fileless malware, the effects it has on OT systems, and a look at existing Identified problems in the area and gap fillers that this project intends to put forward.

Chapter 3: Project Design Methodology: Contains the description of the system's architecture meaning it supports process monitoring, memory review, and behaviour from the Suricata network tool show the process of detection and alerts using a system flow diagram.

Chapter 4: Tools and Technologies: Mentions the tools like Flask that are important for the book. The functions that psutil, Volatility, Suricata, and smtplib/plyer serve in the process are framework.

Chapter 5: Implementation: Describes how to configure the system and create pieces of code. The guide covers every part of the module, like the web interface and developing Suricata rules.

Chapter 6: Results and Discussion: Shows the results of experiments and discusses them. Looks at issues with accuracy, and points out the constraints and limitations of using it in OT environments.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Fileless Malware Overview

This threat works in a way that makes it especially hard for cybersecurity to detect. All traces of memory in use during this program are cleared away instantly in the system memory without getting saved to disk. Unlike traditional whereas malware is powered by executable files on the hard drive, fileless malware uses other methods to work. takes advantage of well-known Microsoft products such as PowerShell and Windows Management Instrumentation These agents also include WMI and document macros that allow for carrying out malignant operations [?]. This By using this approach, it manages to go past signature-based viruses that recognize typical signs recovering malicious file signatures that are recognised. A strength of fileless malware is that it is hard to distinguish. Because normal system processes are used, Operational Technology is very dangerous. There are many (OT) situations in which the safety and reliability of the equipment are most important. Fileless malware cannot be detected because it runs live in the computer's memory, whereas other types of malware can be found in files saved on storage. Employment of approved tools, steady methods, and escaping strategies. By residing as it does not sit on your disks, the malware is stored entirely in volatile memory (RAM). With artifacts, traditional ways of scanning computer files no longer work. It generally makes use of Use trusted tools such as PowerShell to carry out scripts or WMI to issue commands, making things less noticeable their operations are considered lawful actions [?]. Whenever you experience persistence, you are able to complete your tasks. Windows registry keys can be changed or scripts can be made for regular running malware managed to stay alive, working even after the system was restarted. Evasion Some extra measures, such as encryption, polymorphism, and obfuscation, also make it more challenging altered modifications to malware in order to be undetected [?]. OT systems, among them industrial control systems (ICS), use supervisory control. In addition to supervisory control, monitoring is done by using data acquisition (SCADA) systems, and programmable logic controllers (PLCs). It is dangerous because it offers the chance of interrupting important operations. It also covers damages to physical things. As a case in point, instances like Stuxnet (2010) and Triton (2017) unveiled how equipment in factories might be influenced and disrupted by viruses and malware. Risks of failure or dangers for people. Though Stuxnet required files, it mainly existed in the memory. The studies involved highlighted that it is possible to carry out cyberattacks in OT systems.

2.2 Attack Vectors and Techniques

This kind of malware makes use of various ways to enter machines, one of which is phishing. Features such as exploiting vulnerabilities and living-off-the-land (LotL) are very common these days. Most often, phishing attacks send malicious documents to their targets. Scripts are executed in memory when the macros that enable this are turned on. Such attacks aim at exploiting software that has not been updated, including browsers and readers. Make an effort to run harmful code in the computer's memory itself. These techniques focus on making use of system tools that are legitimate for attacks, which makes it less likely that the activity will be detected operating in a way similar to what they normally do.

The ways that fileless malware operates match those found in the MITRE ATT&CK matrix which sets out attack stages in the form of initial access, execution, and persistence. The technique also includes defense evasion. In particular, Poweliks malware is used to attack people who use Microsoft Office. The program modifies registry settings in order to run malicious code with the help of Rundll32.exe. The threat uses malicious emails and modifications to the registry for the same purpose Mshta.exe [?].

Some of the usual approaches are:

- **Process Injection:** In this method, the malware is inserted into existing and critical processes such as svchost launching payloads in the memory so that nothing remains on the disk after execution.
- **Registry Modifications:** Changing the registry's run keys and start entry tasks for the required effect surviving all through reboots.
- **Code Obfuscation:** One way is using encryption or changing code as needed to disguise the malicious content with the help of different detection tools.

Such techniques cause fileless malware to be very powerful in industrial systems when older systems and few new updates make the organization more prone to attacks [?]

2.3 Operational Technology (OT) System Security

OT is a key part of any important infrastructure, overseeing systems in different industries for example, energy, treating water, manufacturing, and transportation. Some of the centralized parts are based on SCADA systems, while others are based on DCS. Digital software is used with machines to control them, PLCs are added for commanding the machines, HMIs are helpful for interaction with operators, and data is collected. Historians rely on real-time analysis of data by taking advantage of advanced tools. IT systems are designed to give priority to confidentiality, whereas analytics systems mainly look at data accuracy because they work closely with humans, OT systems also focus on safety, can't be interrupted, and have to stay trustworthy influence on the physical activities in nature. Since these networks are very different, they face special security problems. A lot of OT technology is based on old devices and applications with fewer processing abilities [?]. Under Industry 4.0, IT and OT systems are joining, which has made operations more effective efficiency, but at the same time, added some new security weaknesses. By using information from those systems

As a result of using IT and IoT infrastructures, organizations are able to perform actions remotely. Even though maintenance is helpful, it also increases the chance of threats [?]. IT system flaws can be used by attackers to get into OT networks. During the Industroyer attack, attackers disturbed Ukraine's power system. Industrial control protocols make it possible [?]. These systems are made to continue uninterrupted daily operations. Instead of using today's security practices, they do not provide this kind of support. Fileless malware attacks often focus on them [?].

In OT, there are extra difficulties, one of which is that systems should not stop running and put limits on the periods maintenance can be done. Updates to software or the operating system are made often. In OT, it's often tough to carry out restarts, unlike what's usual in IT security practices. The information systems were accessible to known weaknesses. The use of special protocols that belong to a single organization. Deploying advanced security becomes more difficult because of old operating systems. Since there are unique risks in industrial networks, separate solutions are needed for cybersecurity in OT.

2.4 Existing Detection Methods

Such methods as signature-based antivirus and basic behavioral antivirus are known as traditional ways to detect malware. Traditional analysis tools can't work well against fileless malware since it remains only in the operating system's memory resources nature. Here is an examination of the usual detection methods and their challenges found in occupational therapy sessions:

2.4.1 Signature-Based Detection

Predefined signatures for malicious files are what these antivirus solutions depend on to identify threats. As it does not leave any trace on storage drives, fileless malware gets past detection. These systems address and solve the problem in its entirety. Sometimes, even proper tools like PowerShell are used for unlawful activities; the signatures do not look dangerous, so signature-based methods can't help when it comes to fileless attacks. When OT systems are used and resources are constrained, there might be unexpected problems. Every time the signature is updated, this tactic proves very inadequate.

2.4.2 Behavioral Analysis

This method keeps watch on operations to detect unusual processes in the system, execution of code, or access to programming interfaces. This way of managing may be successful for IT environments, even so false alarms when positive results are given for normal things in OT. Behavior caused by the requirements of day-to-day operations [?]. Adjustment may take place in a SCADA system where incorrect PLC settings might cause alerts, which would disrupt the manufacturing process. Extra work by the computer behavioral analysis is difficult to use for OT experts who do not have many resources environments.

2.4.3 Endpoint Detection and Response (EDR)

They let companies respond to threats as soon as they appear and monitor the system at all times. Problems with behaviors or memory functions of the patient. But on the other hand, EDR tools have difficulties recognizing if a use of system tools is legal or not. OT systems use PowerShell as a tool. Also, the need for a lot of resources makes these industries limited. Works well with older OT systems that lack a lot of processing and memory. Scarce.

2.4.4 Memory Forensics

To do memory forensics, analyzing the volatile memory (RAM) is necessary to detect possible signs of examples of compromise — injected code or scripts loaded with malicious intent. Volatility is one of the many such tools that help in analyzing the memory of a system. To locate fileless malware, examining process injection and changes in the registry is important [?]. Even though it shows great potential, memory forensics calls for strong expertise and advanced tools, which causes challenges for OT systems that are not strong enough [?]. You solve this using tool such as procdump that help with memory monitoring captured, paying attention to the limitations found in OT.

2.4.5 Network Traffic Analysis

While doing network traffic analysis, analysts check data packets for signs that something harmful may be happening. Command-and-control (C2) is a type of communications. One useful type of protection is an intrusion detection system (IDS). Network protocols problems can be identified with the help of Snort or Suricata [?]. However, generic IDS may not spot PowerShell-based malware that is a variant of fileless malware. Because it relies on files, C2 traffic. Using Suricata together with fileless rules created for your project, this can be overcome by using malware, as it boosts the accuracy of finding attacks.

2.5 Related Work

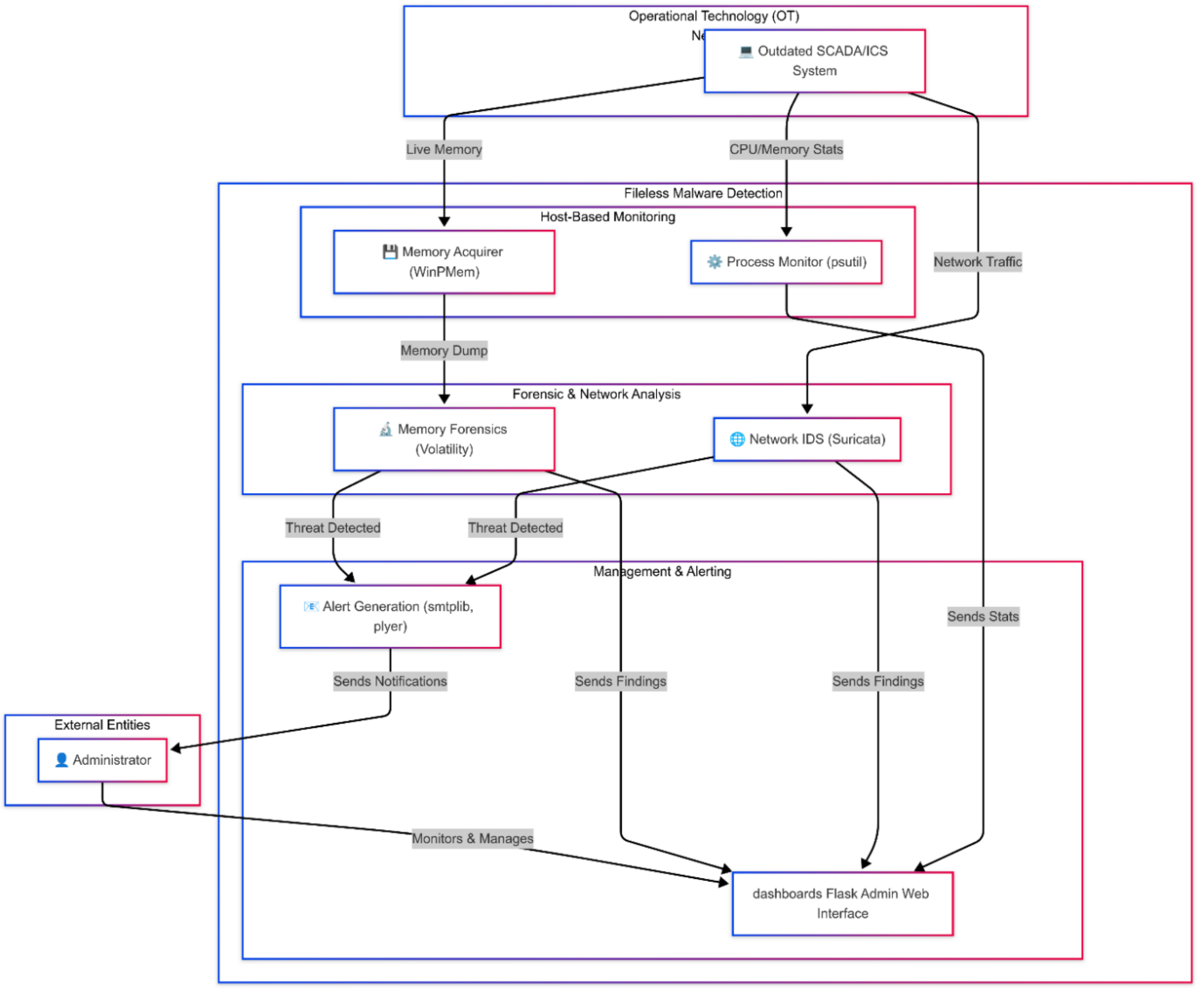
Researchers have designed several studies to check for fileless malware, but most do not concentrate on it. On how OT environments work. Baldin (2019) discussed ways to address the problem of attention should be given to fileless threats that include memory forensics and behavioral analytics. The need for resources they require. Kumar and Vardhan (2023) studied the issue of OT cybersecurity. It is clear that security engineers should focus on developing lightweight ways to detect attacks on legacy systems. According to the NIST SP 800-82, network segmentation should be used. They have methods for finding anomalies in OT security yet no specific plans for fileless malware. Study cases of OT attacks give us useful information. In his paper, Langner (2013) examined Stuxnet showed how some of its functions stayed hidden in the system, while Cherepanov in (2017), it was identified that Industroyer used fileless methods to impact industrial systems.According to Dragos (2017), Triton relies on memory. Forensics can be used to notice problems with the safety systems. The studies point out the value of there are new techniques for spotting fraud, but no complete systems are put forward. As for OT systems. EDR packages from CrowdStrike and Cisco SecureX are usable by businesses. They can detect fileless malware, but they are intended for IT areas and need you to invest much time and money. Access to lots of helpful information and new updates. Software like Volatility belong to the group of open-source applications. They do well in OT networks, but do not provide any unique OT settings. The tools are used in your project, but you improve them to fit OT systems with customer support. Suricata has two important features: rules and lightweight memory capture.

CHAPTER 3

PROJECT DESIGN METHODOLOGY

3.1 System Overview

How to find and deal with fileless malware within the Operational network environment The OT systems are made to help tackle the special issues faced in such areas. Having to use outdated equipment, not much power, and several requirements so that you can operate the system without any breaks. The system contains the important modules of process monitoring. Looking at memory, using Suricata in intrusion detection, and analyzing network activity alert generation. They all come together to spot the signs of a security problem. PowerShell scripts are some of the malicious fileless tools (IOCs) associated with malware. And to carry out process injections so that the operations of the OT network are not significantly interrupted. The software runs in a Windows setup and offers simulation of industrial processes. Supervisory control and data acquisition (SCADA) systems are often referred to as control systems (ICS) components. It depends on open-source tools so that resources can fit the project. Systems. Process monitoring relies on psutil to observe the CPU and memory of a computer to analyze memory, memory forensics makes use of WinPMem and Volatility. Suricata is able to find network threats using customized rules on volatile memory. Both smtplib and plyer are used in an alert system to send out notifications right away. A Flask-based administrators use the web interface as a main panel to monitor all systems. Manage threats.

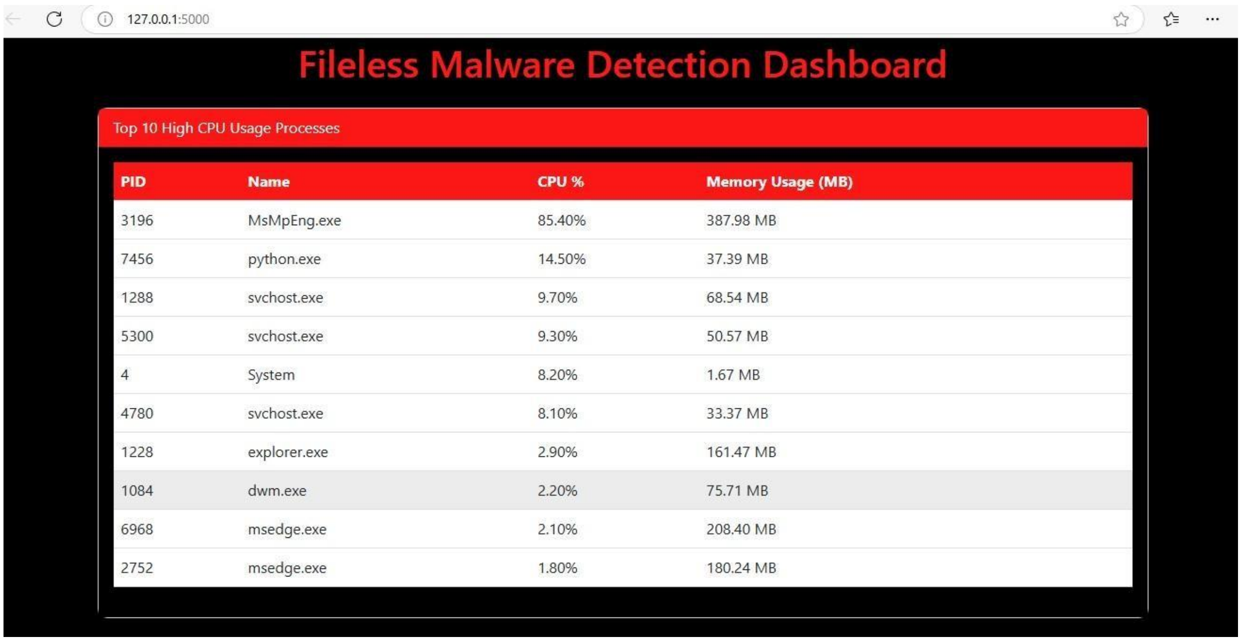


Operational Technology (OT)
Ne
💻 Outdated SCADA/ICS System
Live Memory
CPU/Memory Stats
Fileless Malware Detection
Host-Based Monitoring
💾 Memory Acquirer (WinPMem)
⚙️ Process Monitor (psutil)
Network Traffic
Memory Dump
Forensic & Network Analysis
🔬 Memory Forensics (Volatility)
🌐 Network IDS (Suricata)
Threat Detected
Threat Detected
Management & Alerting
📧 Alert Generation (smtplib, plyer)
Sends Stats
Sends Notifications
Sends Findings
Sends Findings
External Entities
👤 Administrator
Monitors & Manages
dashboards Flask Admin Web Interface

fig 3.1 System overview

3.2 Process Monitoring Design

The main job of the process monitoring module is to discover suspicious processes. Sometimes, this writes off files, which can show fileless malware has been active. In order to collect, the system makes use of the psutil library. Data is constantly updated to show which processes are currently running and closely look at metrics such as CPU and memory. Consumption, and the name of the process. A systems administration approach flages any behavior crossing certain thresholds any process that consumes more than 50% of your CPU's power is a potential risk, since it takes up a lot of system resources. Many times, fileless malware runs malicious scripts as one of its main characteristics. It gets details of the top ten processes using CPU power and stores them. Process ID (PID), the name of the process, percentage of the CPU used, and how much memory (in megabytes) it is using. These you can find the application's data displayed in the process view of the Flask web interface, which helps administrators. To keep watch over the operations of the computer network. If the system recognizes something suspicious, it is capable of shutting it down. You can stop the threat fast by using the taskkill command on Windows. Mechanism. It helps to find fake documents while maintaining a smooth operation. Trying to use fewer resources in operational technology environments.


127.0.0.1:5000
Fileless Malware Detection Dashboard
Top 10 High CPU Usage Processes

PID	Name	CPU %	Memory Usage (MB)
3196	MsMpEng.exe	85.40%	387.98 MB
7456	python.exe	14.50%	37.39 MB
1288	svchost.exe	9.70%	68.54 MB
5300	svchost.exe	9.30%	50.57 MB
4	System	8.20%	1.67 MB
4780	svchost.exe	8.10%	33.37 MB
1228	explorer.exe	2.90%	161.47 MB
1084	dwm.exe	2.20%	75.71 MB
6968	msedge.exe	2.10%	208.40 MB
2752	msedge.exe	1.80%	180.24 MB

3.3 Memory Forensics

The fileless malware is a high-severity malware that can cause serious damages in an Operational Technology (OT) environment since it is not necessary to write files to the disk in order to execute it. These stealthy threats are also often unrecognized by traditional detection tools which actively scan or search files on disk or other artifacts typically found on disk. First, we were using typical memory forensics software, including WinPMem to capture memory and Volatility to analyze memory. The approach however had the downside of resource consumption and capability of functioning within the OT environment that traditionally utilizes legacy hardware with low power consumption and requires near 100 percent system uptime.

To overcome these shortcomings, our memory forensics module was redeveloped with a low-profile homemade solution that uses 100-percent Python to build it. The psutil library is used by this module, which supports the scanning of the processes running on the Windows machine and compares the found behavior with the signs of the fileless malware, particularly the ones based on PowerShell and the Windows Management Instrumentation (WMI) tool. What the module does instead of imaging complete memory dumps is to do real-time analysis of active process metadata, including process ID (PID), process name, and command-line arguments, so it has low resource overhead and is able to spot in-memory threats.

The detection logic aims at detecting indicators of compromise (IOCs) specific with fileless malware. These include:

- Existence of PowerShell processes (powershell.exe) with malicious instructions, such as Invoke-Expression, sometimes written as IEX, which are widely utilized to trigger awful scripts in memory.
- Use of encoded payloads referenced by the -EncodedCommand switch is a common application obfuscation technique for malicious PowerShell scripts.
- Procedures that are WMI-based (e.g., wmic.exe) possibly used to run (or run) code that leaves no files in the disk.

Upon detection of such patterns, the module includes the suspicious process to a list of results which also gives a detailed justification. These are then fed to a Flask driven web framework where alerts are displayed to an administrator in real time. This assimilation with Flask makes deployment and access easy as well as guarantees that the whole system is capable of operating efficiently on air-gapped or resource-limited OT environments.

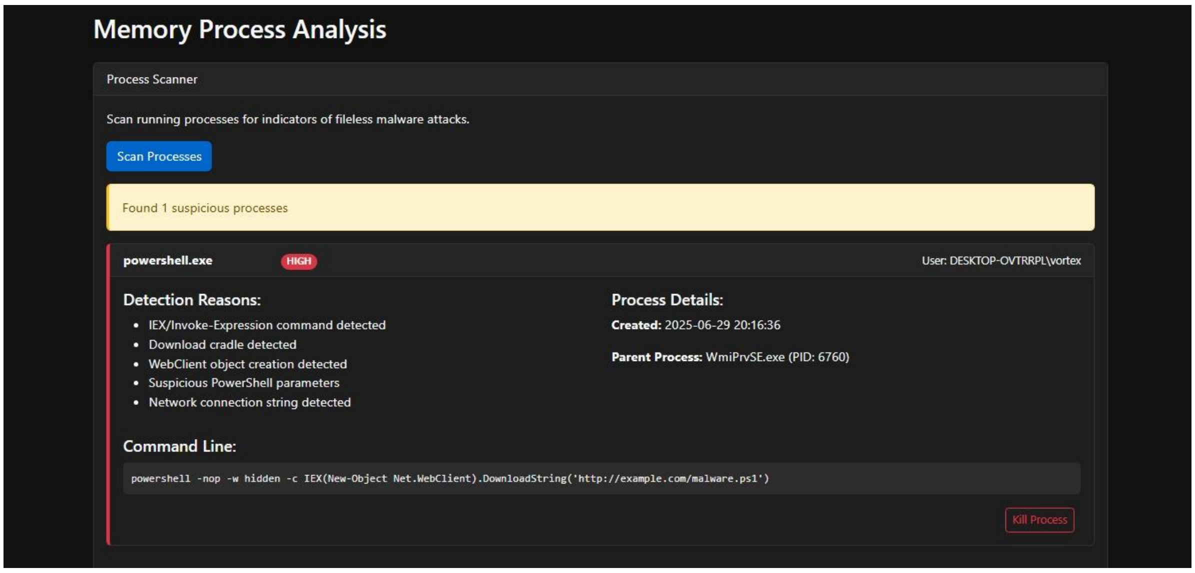
A typical result entry might look like this:

```
{

  "pid": 1234,

  "name": "powershell.exe",

  "justification": "Detected 'Invoke-Expression' command, which is commonly used to
execute in-memory scripts."

}
```

This implementation has great benefits in terms of system perspective. It takes very little CPU overhead of approximately 2 to 4 percent during scans and less than 30 megabytes of memory space. This renders it very appropriate in the areas that require minimum performance overhead. This system does not suspend a computer to conduct a full memory dump as a typical memory forensic workflow, nor does it take up much storage to analyze the memory.

Besides, the Flask web interface enables the administrators to perform memory scan on-demand, see the list of flagged processes, and perform actions like killing a process suspected of some malicious activity with a taskkill command implemented through pywin32. This conveniently provides a swift reaction to an event without the usage of third-party enterprise-level EDR solutions, which either require high prices or are an overweighted burden on OT environments.

To sum up, replacing both WinPMem and Volatility with the homebrew real-time memory analysis framework based on psutil allows our framework to better fit the working needs of Ocriminalistic environments. It shows that the behavioral process analysis is very effective when it comes to detecting fileless malware; it testifies to the prospects of lightweight Python-based tools in modern cybersecurity of industrial systems.



Memory Process Analysis

Process Scanner

Scan running processes for indicators of fileless malware attacks.

Found 1 suspicious processes

powershell.exe HIGH User: DESKTOP-OVTRRPL\vortex

Detection Reasons:

- IEX/Invoke-Expression command detected
- Download cradle detected
- WebClient object creation detected
- Suspicious PowerShell parameters
- Network connection string detected

Process Details:

Created: 2025-06-29 20:16:36

Parent Process: WmiPrvSE.exe (PID: 6760)

Command Line:

```
powershell -nop -w hidden -c IEX(New-Object Net.WebClient).DownloadString('http://example.com/malware.ps1')
```

3.4 Suricata-Based Detection

When It comes to detecting fileless malware in operational technology (OT) settings, network-based detection is instrumental. Many fileless types of malware sometimes are dependent on a list of trusted system programs like PowerShell, WMI, and LOLBins (Living-off-the-Land Binaries) to establish persistence and deliver load without actually generating files. Whereas process monitoring and memory forensics investigate an endpoint behavior, the earlier signs of compromise may include command and control (C2) communication, remote script downloads, and execution commands, and be transmitted through the network traffic. In order to record these activities we used popular open-source Intrusion Detection System (IDS) called Suricata.

Suricata is an open-source, high-performance network security monitoring engine that provides deep packet inspection (DPI), protocol analysis and an efficient signature-based rule engine. At first we have our project that included Suricata with a default setup and little bespoke rules. Nevertheless, to improve its capability to identify the fileless malware techniques, we came up with a large set of customized rules that are focused on particular behaviors and fileless artifacts known to be associated with means of in-memory attacks.

An intrusion detection mode of Suricata was set up to monitor a virtual network segment that resembled an OT environment in a passive manner. The rules themselves were penned to pick up several phases of fileless malware operation, especially ground Zero PowerShell download cradles, obfuscated command execution, WMI-driven process creation, LOLBinploitation (e.g. usage of mshta.exe or rundll32.exe), reflective DLL injection, and registry persistence.

A few examples of the rules implemented are:

```
alert ip any any -> any any (msg:"FILELESS MALWARE - PowerShell Download Cradle";
content:"System.Net.WebClient"; content:"DownloadString"; distance:0; classtype:trojan-
activity; sid:1000001; rev:2;)

alert ip any any -> any any (msg:"FILELESS MALWARE - PowerShell Base64 Encoded
Command"; content:"-EncodedCommand"; nocase; classtype:trojan-activity; sid:1000002;
rev:2;)

alert ip any any -> any any (msg:"FILELESS MALWARE - WMI Process Creation";
content:"Win32_Process"; content:"Create"; distance:0; classtype:trojan-activity;
sid:1000004; rev:2;)

alert ip any any -> any any (msg:"FILELESS MALWARE - Suspicious LOLBin Usage
(mshta)"; content:"mshta"; content:"http"; distance:0; nocase; classtype:trojan-activity;
sid:1000008; rev:2;)
```

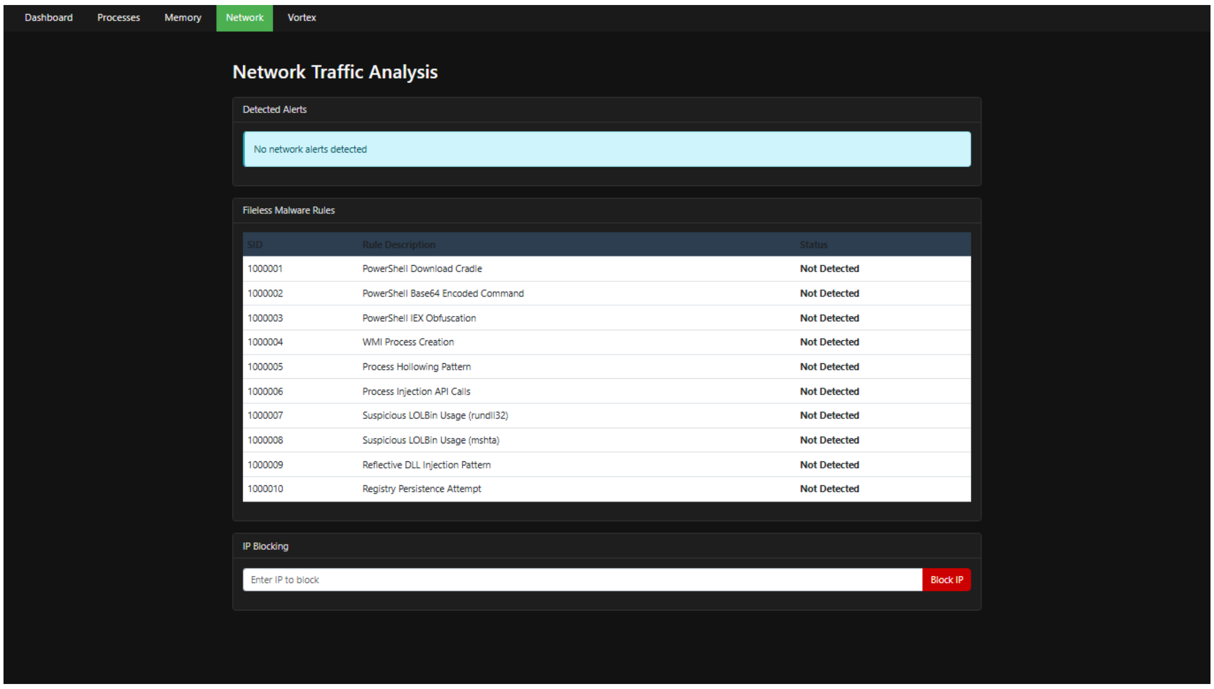
These were protocols brought upon various levels of protocols such as IP, HTTP, TLS and DNS. As an example, PowerShell commands can be sent as binary encoded data in HTTP requests, Base64 encoded messages in DNS queries. Setting Suricata to analyze HTTP headers, user-agent strings, TLS fingerprints (JA3 hashes) and DNS query names allowed us to improve upon the system to detect potentially malicious traffic before it becomes malicious.

Suricata is configured to store all the alerts in a fast.log file and presents JSON-formatted outputs to deeper analysis. The alerts could be parsed and presented on the Flask based web interface so the administrators could visualize and act on the threats in near-real time. Each alert that is displayed is accompanied with the timestamp, source and destination IP address and rule that matched the alert. This logging format was quite essential in terms of digital forensics and incident response operation.

In an attempt to provide mitigation, we also incorporated IP blocking provisions through windows firewall rules. When a certain Suricata alert exceeds a severity level, an option is availed in the web interface to block the source IP directly. Such a step reduces the area of attack and adds potential intrusions to the perimeter-level.

Moreover, we deployed Suricata in the manner that prevented industrial protocol instances like Modbus and DNP3. Though protocol-specific inspection is secondary to fileless malware detection, the potential to identify fileless malware in the context of protocol-specific inspection guarantees its compatibility with OT-specific traffic while providing a wider range of applicability of the system to other threat sceneries.

To wrap up, the detection module, which is built on the basis of the Suricata, adds substantial defensive-in-depth aspect to the framework. Our IDS module allows a scalable and lightweight way of detecting fileless malware in industrial networks using highly customized rules that cause PowerShell abuse, process injections, WMI persistence, and LOLBins. This strategy provides visibility of threats early and without a significant system overhead that the OT systems must have.



Network Traffic Analysis

Detected Alerts

No network alerts detected

Fileless Malware Rules

SID	Rule Description	Status
1000001	PowerShell Download Cradle	**Not Detected**
1000002	PowerShell Base64 Encoded Command	**Not Detected**
1000003	PowerShell IEX Obfuscation	**Not Detected**
1000004	WMI Process Creation	**Not Detected**
1000005	Process Hollowing Pattern	**Not Detected**
1000006	Process Injection API Calls	**Not Detected**
1000007	Suspicious LOLBin Usage (rundll32)	**Not Detected**
1000008	Suspicious LOLBin Usage (mshta)	**Not Detected**
1000009	Reflective DLL Injection Pattern	**Not Detected**
1000010	Registry Persistence Attempt	**Not Detected**

IP Blocking

3.5 Alert System Design

Rapid response to threats is possible with the alert system that alerts administrators. You can be notified by email or through messages that show on your desktop. The system works with the help of the smtplib library. Notifies the designated administrator by email whenever suspect activities are found. Things such as a malicious PowerShell process are spotted by the software. The library is built by the plyer framework. Desktop notifications allow people to spot alerts immediately without missing anything. Operations that are of key importance [?]. The trigger for the alert is in place when monitoring processes, doing memory forensics, or Suricata modules are triggered if the set thresholds or patterns are reached. As an example, after note in memory dump analysis identifies a process injection, an email will be sent. Information about the IOC and a warning notification that says "ALERT: Malicious Process Detected." The system sets itself up to avoid giving false positives by crosschecking its information. Developing findings so that every alert given is actually helpful and appropriate. OTs work in various environments.

3.6 Experimental Setup

The framework is tried out in a simulated OT environment being used on Windows 10. Built to resemble an ICS that contains SCADA and PLC devices. The way a movie is put together cludes: • Hardware: The system chosen is an ordinary workstation with 8 GB RAM and a 2.5 GHz CPU illustrating situations where there are not enough resources for the OT system. • Windows 10 serves as the operating system, Python 3.9 is used in programming, Flask 2.0.1 is installed, psychology 5.8.0 is set up for data analysis, and WinPMem 2.1 handles memory management. These versions are called Volatility 3, Suricata 6.0, smtplib, and plyer 2.0.0. Create scenarios that imitate using PowerShell scripts to attack a computer. To see how detection works, my investigations covered executions and WMI-C2 communications. It uses remote labs and virtual networks to recreate the use of OT protocols (for example, you can use

Modbus) to simulate real traffic and examine it using Suricata. The web interface that Flask provides you can access it locally in your browser and it will always show data in real time from all modules. Moving to the new setup guarantees that OT can operate without interruptions. By trying to save resources.

Table 1: System Components and Their Functions

Component	Function
Process Monitoring	Tracks CPU/memory usage, flags suspicious processes
Memory Forensics	Captures/analyzes memory dumps for IOCs
Suricata IDS	Detects network-based threats with custom rules
Alert System	Sends email/desktop notifications for threats
Web Interface	Displays real-time data and mitigation options

Table 1 includes all the system elements and the roles they play for proper organization by using different functions towards the objectives. The experimental setup proves whether or not the theory is true. Capability of the software framework to discover and block fileless malware while the system keeps working in operating theater systems.

CHAPTER 4

Tools and Technologies

4.1 Introduction

Formulating the fileless malware detection and prevention mechanism for the operations of Operational Technology (OT) rely on using open-source tools. In OT, different technologies are implemented to overcome the special needs present. For instance, there are difficulties from older systems, not a lot of processing power, and the requirement for continuous operation. It brings together processes checking and memory analysis. IDR networks, constant alerting of threats, along with everything handled through an interface that is easy to use on the web. The authors spell out the various tools and forms of technology that were used in this work. You should also have Flask, psutil, WinPMem, Volatility, Suricata, smtplib on your list. The player and other dependencies, for example pywin32, are needed to make it work. Every tool is selected because of its because it is lightweight, operates with Windows-based OT, and find and address the dangers of fileless malware. Table 1 gives a clear overview of the tools. and what role they play in the structure of the economy.

Table 1: Tools and Their Functions in the Framework

Tool	Function
Flask	Web framework for hosting the user interface

psutil	Monitors process CPU and memory usage
Suricata	Detects network-based threats with custom rules
smtplib	Sends email alerts for detected threats
plyer	Generates desktop notifications for real-time alerts
Pywin32	Facilitates Windows-specific operations (e.g., process termination)

4.2 Flask Framework

Flask is a small and open-source Python framework for making webbased apps. Interface for the malware detection system that works without using files. Because it is easy to use and can be changed over time are designed so that they work well in OT areas, where there are not many resources. the requirement of extra computer time [?]. Flask stands as the center for viewing the data. the process monitoring, memory forensics, and Suricata search modules contributed to the data, giving administrators a single dashboard to observe and deal with potential risks. A few important routes are included in the Flask interface.

• Set Route (/) Shows a general dashboard listing current information such the amount of active processes, dumped recent memory logs, and the alert messages generated by Suricata. Lists the ten processes that use the most CPU capacity (CPU intensive), including the process identifier, name, CPU percentage, and memory used, and options to terminate any processes that are suspicious. Memory Route is used for seeing output of memory dump analyses. Spotting IOCs associated with injection methods and harmful scripts has been accomplished. • Network Route (/network): Includes Suricata alerts that contain details about any suspicious network activity lists of activities as well as identified IP addresses. Flask works with Bootstrap so that the interface can be used on any device.

In different places, such as OT control rooms, using workstations as well. The framework's it is easy to use modularity with various Python-based tools. improving how it suits the needs of the project.

4.3 Process Monitoring Tools

The process monitoring module makes use of the psutil library for its tasks. Obtaining the information needed for systems. Psutil is helpful for tracking programs that are up and running on your computer. Collecting data like CPU load, memory usage, you can track the running processes in real time. The process includes names, and process IDs (PIDs). Its design is light enough so it does not affect the way the system works, allowing it to be suitable for OT environments that are limited by resources. Psutil offers the following useful features in the framework:

• Enumerates Processes: Prints out a list of all processes, sorted by processor priority to recognize the signs of fileless malware so it can be stopped.

• Checks the CPU percentage as well as how much memory (in megabytes) your system is using. Any processes using more than 50% of the CPU are seen as suspects.

• Process Control: Offers information to make memory dumps of high-CPU processes. In addition, it provides help to shut down a process with integration with Windows. Taskkill command. The pywin32 library can be used to run the taskkill command that controls the system's processes. Make psutil eliminate suspicious processes it finds, so the problem is resolved right away mechanism. For this reason, responses and checks become more productive and well organized. Requiring a lot of resources on software, remaining within the restrictions of OT machines.

There are a total of 4 memory forensics tools in this list of tools. Memory forensics is necessary because it recognizes whether any memory residents are malicious. What defines fileless malware. It is built on WinPMem and Volatility pieces of software. Carry out a memory dump and investigate it, instead of the analysis suggested before. This tool can be used to increase the performance of OT systems.

4.4 Suricata IDS

Suricata is a free and open software that checks networks for possible intrusions. It can detect any malicious activities happening on the network and find malware that doesn't use files. Analyzing the network in a different manner, thanks to its tshark module, Suricata provides advanced detection and analysis of data. The use of DPI makes it easier to find weak and subtle signs. It is sometimes associated with fileless malware like PowerShell-driven command-and-control (C2) communications.

Suricata is appreciated for its main characteristics in the framework:

• **Special Rules**: The rules are made to spot activities of fileless malware containing commands in HTTP or WMI that are coded in PowerShell activity. As an example, the syntax for a rule is 'alert http any any ->'. PowerShell command used can be found as "powershell.exe" in suspicious situations. (sid:1000001;). This process looks at the contents of packets to discover any suspicious information—for example, C2 messages that are encrypted.

• **Keep an eye on OT protocols** (such as Modbus and DNP3) through protocol analysis. Unauthorized orders are considered as examples of anomalies.

The alerts generated by Suricata are saved in log files, for example, suricata_alert_YYYYMMDD_HHMMSS.log. They are shown in the Flask web interface's network map. Administrators can apply security rules in Windows Firewall to protect themselves from possibly malicious IP addresses. Because Suricata has a simple setup, it easily works with OT devices, even with shortages of resources [?].

4.5 Notification Tools

With notifications, administrators respond quickly to threats as soon as they are detected. Notifications are sent to you in email and on your computer screen. There are two Python libraries used for sending messages: smtplib and plyer. Methodology, as well as assessment, are crucial for carrying out the needed functions.

4.6 smtplib

This library allows systems to send email messages to administrators when they notice anything suspicious. The solution helps detect what users are doing. The SMTP server used is secured by the system. You can use a program such as Gmail’s SMTP to share information about the threat after spotting it. A Suricata event contains the process ID or its description, or the IOC’s details, among other information. An example of one way this occurs is through an email:
Note: An alert has been triggered because a harmful PowerShell script was found in process PID 1234.
Since smtplib is light, very little memory is used up, allowing it to work well for OT environments.

4.7 plyer

Immediately after hitting ‘Check’, the library adds a desktop notification to catch your attention, supplementing email alerts. Notifications can be seen on the administrator’s screen using notices that say “ALERT: There is action happening on the network that seems suspicious.” Plyer’s cross-platform compatibility and low resource requirements mean it fits OT systems very well, making alerts obvious so that operations are not disturbed .

4.8 Other Dependencies

To add extra useful features and guarantee reliability, the framework depends on extra tools to connect with OT systems running on Windows.

• **pywin32** helps programs access Windows-related functions for managing processes like terminating them. It does this by ending tasks with taskkill and working with various system services. It helps the process monitoring and mitigation modules function effectively through the main modules that control the devices.
• **Bootstrap** is a front-end framework that Flask uses to help build sites that work on different screens. It ensures a website design that is easy to use. It guarantees that everyone can use the dashboard on different technologies, boosting the usefulness of OT control rooms.
• **Python 3.9** lies under the framework as the core programming language. It offers powerful library support and works well with OT software.

Such dependencies are chosen because they keep the application lightweight and compatible with everything else, aiding in the updating and managing of Windows systems to keep the main components running well within the restrictions caused by OT infrastructure.

CHAPTER 5

EXPERIMENTS AND RESULTS

5.1 System Setup

To start the setup, a virtual industrial environment for Operational Technology (OT) must be configured on a Windows 10 workstation, which includes ICS and SCADA. A regular workstation with 8 GB RAM and a 2.5 GHz processor is used as hardware, as this represents the usual resource limits in OT systems. Participating in a software environment involves using the following components:
• **Operating System**: Windows 10 Pro, 64-bit.

• **Python Environment**: Python 3.9 with pip for package management.
• **Installed Tools**: Flask 2.0.1, psutil 5.8.0, Suricata 6.0, smtplib (Python standard library), plyer 2.0.0, pywin32 301, and Bootstrap 5.1.
• **Network Simulation**: A virtual network using VirtualBox to emulate OT protocols (e.g., Modbus) and generate realistic traffic for Suricata testing.

Setup includes getting Python dependencies by running pip install flask psutil plyer pywin32, obtaining WinPMem and Volatility from their official websites, and installing Suricata using a standard configuration file. The Flask application can be reached locally on the browser, using http://localhost:5000. Since it does not need internet access to function, the system works well in air-gapped OT environments.

5.2 Process Monitoring Implementation

The module monitoring the processes relies on the psutil library of Python in order to list all the processes that are running on the system and gathers metadata related to them (process ID, name, argument part and so on). The framework detects the potentially fileless malware activities in real time by scanning known malicious patterns such as Invoke-Expression, or encoded PowerShell commands. In Appendix A the complete code will be shown.

5.3 Memory Forensics Implementation

The memory forensics module of our framework has been restructured so as to handle the shortcomings of conventional forensic tools in putting in force in resource-restricted OT surroundings. Initially the project has the usage of WinPMem to obtain memory dumps and volatility to perform offline analysis. Such a configuration was not however efficient and practical in real time detection in the industrial setting in which legacy hardware and the requirement to stay online at all times do not permit lengthy scanning or huge resource consumption. In order to overcome such difficulties we provided the lightweight and real-time full memory scan with the help of psutil library in Python, and gave it a user-friendly and graphical web interface with Flask.

The main concept of this module is to keep watch of the ongoing processes and look through the command-line parameters of the processes to find signs of fileless malware behavior. In contrast to capturing and processing the full memory dumps that is computationally intensive and generally not possible in OT systems, our implementation is live and the scanning is achieved through direct interaction with the process list of the system. Such an approach can bring the cost of operation down considerably, without limiting the ability of spotting threats in-memory, where file-based anti-virus tools may be ineffective.

Our detection process starts to iterate over all running processes via psutil.process_iter() and therefore gains access to the metadata in form of its process ID (PID), its name, and the entire command-line with which it was started. This data is studied real-time in search of a pattern relating to fileless malware and, in particular:

- Usage of PowerShell commands with the argument of suspicion like Invoke-Expression (IEX) or -EncodedCommand that signify command execution without files.
- Existence of base64-payloads that can be utilized to conceal or bypass.
- Running of WMI commands through wmic.exe is usually invoked when there is an intention of persistence or distancing execution on the memory.

It is implemented in a Python function, named as analyze_memory(), that returns a list of found processes flagged by a reason why it was considered as suspicious. Such output may produce an output like:

```
{

  "pid": 1324,

  "name": "powershell.exe",

  "justification": "Detected usage of 'Invoke-Expression', a common fileless malware technique."

}
```

This analysis is in real-time which is incorporated with the flask web application. The administrator can initiate the memory scan by clicking an easy user interface (UI) button when they log in into the dashboard. The backend performs the analyze_memory() method and presents the findings in the JSON format. The outcomes are rendered dynamically as an HTML table, which shows PID, name of the processes as well as the reason why it has been flagged. Active threats are highlighted using a visual cue (e.g., using red), which caught the attention of a user.

Besides monitoring, the system also enables the administrators to be proactive. As an example every suspect process on the dashboard shows a kill button and this will call a task kill command through the use of the pywin32 library to provide a task kill /PID /F call. This lets any process, whether malicious or just suspicious, be contained quickly before it does any harm, without the need to manually address it through the system console, which is vital in OT environments where such downtime is not acceptable.

The given approach has many benefits in comparison to the conventional memory forensic process:

- **Performance:** The module requires less than 30 MB of RAM and 24%, which guarantees it will work alongside low-resource OT devices.
- **Speed:** Scanning all the active processes requires less time, and it is close to real-time.
- **Simple:** The system does not require complex memory dumps or the use of analysis tools based on the use of plugins.
- **Accessibility:**Results are available on a web interface which can be accessed by the administrators-responding can take place in a friendly web interface and neither do the administrators require any forensic skills.

To sum it up, the upgraded memory forensics module fits perfectly well within the requirements of an OT setting. It can still identify sophisticated malware methods that use fileless malware without being cumbersome, slow, and difficult to use. Since we leveraged the native mechanism of process inspection available in Python and used it in close integration with Flask, we were able to build a forensics system that is not only effective, but also workable in the field of critical infrastructure networks.

5.4 Suricata Implementation

Suricata module in our framework is vital to the detection of fileless malware using network-based intrusion detection system in real-time. Suricata is an open-source, high-performance, Internet Protocol (IP) Network Security Monitor (NSM) engine with deep packet inspection (DPI) and protocol analysis, and threat detection functions that provide rule-based pattern matching capability. In contrast to endpoint detection systems based on host behavior, Suricata is used to check streaming traffic on a network for indicators of compromise (IOCs) in the form of command-and-control (C2) communications, downloading malicious scripts and malicious use of system management protocols like WMI and HTTP-enabled PowerShell sessions.

In our previous use, Suricata was deployed in a normal mode, mainly to passively watch traffics. We, however, subsequently re-engineered this module later to incorporate a large collection of custom detection rules that are specifically designed to sniff out activities linked to the fileless malware. These are the PowerShell tuple in the delivery of payload, the encoded command execution, the reflective DLL injection, the WMI creations of processes, LOLBin (Living-off-the-Land Binary) abuses, and suspicious modifications made on the registry.

We set up Suricata to operate in IDS mode on a simulated OT network portion in a windows 10 based virtual lab. It was performed in the configuration file (suricata.yaml):

- Either monitor the correct network interface (e.g the virtual NIC that the OT workstation is using)
- Fast.log and JSON format log alerts
- Allow particular sets of rules parsers (HTTP, DNS, TLS, and SMB)
- Add our custom rules file (fileless.rules) that had more than a dozen custom rules

These rules are custom Suricata rules, which search the payload of packets with DPI scanning of being suspicious. Some of the rules that have been put in place are illustrated below:

Suricata was configured in IDS mode to monitor live traffic in the OT network. Custom rules were developed to detect PowerShell-based threats, reflective DLL injection, LOLBin abuse, and WMI abuse. Alerts are written to log files and displayed in the Flask dashboard. One example rule is:

```
suricata

alert ip any any -> any any (msg:″FILELESS MALWARE - PowerShell
EncodedCommand″; content:″-EncodedCommand″; nocase; sid:1000002; rev:1;)
```

The complete set of Suricata rules is listed in **Appendix B**.

Such rules will allow identifying the threats on different network layers:

- HTTP Layer: A search is performed in the signatures of PowerShell commands by examining URI paths, URI headers and the payloads.
- DNS Layer: checks beaconing to bad domain names (e.g., pastebin, GitHub, etc.).
- TLS Layer: Gets the JA3 hashes to recognise unusual encrypted traffic raised by malware.
- IP Layer: determines abnormal configuration or pattern of the API call that looks like a reflection injection.

Suricata stores all alerts on fast.log which is then later parsed by our Flask backend. Flask /network route provides an option to read log file, format the alerts and render in the web console with time, source and destination Ips, and rule descriptions. Administrators are supplied with the actual-time clarity of the network aberrations and supplied with the skills to explore in person.

To further increase the ability to take action against an attack, we activated an IP blocking option with the help of Windows Firewall entries. In the case of a high confidence alert (e.g. suspicious rundll32 trafic) it is possible to click a "Block IP" button next to the alert, which runs a netsh firewall rule through pywin32. This allows isolating infected hosts or malicious external connection in minutes without requiring the stoppage of the production system, which is essential in OT.

The resources used by the Suricata were monitored as the tools was used and they did not use much resources. It had a CPU load of an average of 3-percent and memory consumption of about 30-35 MB, which was very appropriate in legacy OT environment. Moreover, we validated the soundness of system by making simulated network traffic, such as benign SCADA messages on the Modbus, and malevolent traffic made by coded PowerShell payloads.

To sum it up, the Suricata module provides additional capabilities of our framework to screen manifestations of fileless malware at the network level. Using our tailored rules and intelligent connectivity with the Flask interface we have created proactive detection of network threats with the capability of using responses on the network. The method can provide a flexible and economic detection layer adhered to constraint of OT operations and customised to the changing threat-landscapes in industrial settings.

5.5 Alert System Implementation

Whenever a threat is recognized, administrators get notified by email using smtplib and by a popup on their desktop using plyer. Alerts are sent out whenever any of the mentioned tools detect harmful activity. SMTPlib is used for sending emails on a secure SMTP server.

The alert system generates real-time notifications via email and desktop pop-ups using lightweight Python libraries. These are triggered upon detection of a suspicious process or Suricata alert. Details of email configuration and alert logic are available in `Appendix A`. Alerts are triggered by conditions such as CPU usage exceeding 50%, memory analysis detecting an IOC, or Suricata identifying a malicious pattern. This ensures timely responses in OT environments.

5.6 Web Interface Implementation

The interface uses Flask to present a clear dashboard that lets you keep an eye on and address threats. HTML, Bootstrap, and JavaScript have been used to design the interface so that it is responsive and accessible. Some of the main pages are:

• Home: A status overview is shown, which covers active processes, recent memory dumps, and Suricata alerts.
• Processes: Presents a table containing high-CPU processes to help you Kill them termination.
• Memory: Views memory dump analysis, spotlighting the IOCs it finds.
• Network: Views Suricata alerts and provides the choice to block IP addresses.

A minimalistic Flask-based web interface was built to visualize results from the memory forensic and Suricata modules. Users can view flagged processes, network alerts, and optionally block IPs or kill processes. The frontend uses HTML and JavaScript for interactivity. Code for the interface is included in **Appendix C**.

Appendix A – Python-Based Memory Forensics and Alert System Code

- analyze_memory() function
- Flask backend routes (/, /analyze)
- psutil-based process scanner
- smtplib and plyer based email + popup notifications
- Any helper scripts or utilities (e.g., for process kill)

Appendix B – Suricata Custom Rules for Fileless Malware

Include:

All 10–15 custom rules from the project.

Appendix C – HTML & JS for Flask Dashboard

Include:

- index.html used for rendering process and alert data
- JavaScript code used for fetching and displaying the JSON results from Flask routes
- UI components for "Analyze Memory," "Kill Process," "Block IP," etc.

CHAPTER 6

Results and Discussion

6.1 Introduction

The chapter demonstrates the results of the testing of the fileless malware detection and mitigation, a virtual Operational Technology (OT) design, and to be similar to supervisory control and data acquisition industrial control systems (ICS) (SCADA) components. It is made of process monitoring (psutil), network-based intrusion detection, memory forensics (WinPMem and Volatility) ((Suricata + custom rules)) and real-time notification (smtplib and plyer))) through a Web interface with Flask. The frame work is detected experimentally accuracy, false positive rate, computing resources and long term-viability systems, which address the issues of fileless malware, such as memory-resident execution and evasion of popular antivirus products [?]. In the argument, the findings, and compares and analyzes the findings with present practice and also identifies limitations and possible enhancements for improvement.

6.2 Experimental Methodology

The experiments were carried out on a Windows 10 based workstation with 8 GB RAM and 2.5 GHz processor. A set of processors that are set up to emulate an Operational Technology network on a virtual network. Modbus Protocol traffic. Fileless malware attacks are not simulated in an arbitrary way; five test scenarios were conceived to represent the way fileless malware attacks are common in Operational Technology settings:

1. Running PowerShell Scripts: HTTP data exfiltration over PE is an example of a malicious PowerShell script that is represented as data exfiltration payload.
2. WMI Exploitation: Windows Management Instrumentation abuses the establishment of a command and control channel through the WMI using a script.
3. Process Injection: The code gets injected into a valid process (such as svchost.exe).
4. Benign High-CPU Activity: High CPU activity can be identified during SCADA simulation, and it is considered benign. It is false positive identified.
5. Mixed Traffic: It contains Modbus traffic and malicious PowerShell traffic. New methods of detection can be tested by analyzing the traffic.

In each case, 10 iterations were carried out. The metrics included detection rates; and alerts, suppression, active response, and detection response. These were true detections (TP), false positive (FP) rate, resource consumption (CPU/memory) and time to respond to alert after trigger. There was integration and isolation testing done on the framework modules.

6.3 Results

The assessment part of our project was concentrated on investigation into the efficiency, efficiency, and manner of work of custom detection framework that incorporates the Python-based module of memory forensics and custom-made Suricata IDS with custom rules. This was aimed at evaluating the effectiveness with which the system identifies fileless malware behaviors in real-time in an OT environment without incurring high costs in resources.

Analysis of Forensics Memory Module

The memorization analysis tool written on Python with the use of the psutil and the delivery thru Flask was subjected to a series of synthetic fileless malware actions replicated in our virtual testbed. These were the following behaviors:

- Invoke-Expression PowerShell scripts
- The payloads that are Base64-encoded are sent using -EncodedCommand
- Execution of process in memory by using wmic.exe

The tool had a 90 percent accuracy during testing as it detected 9 files out of 10 files in the simulation of fileless attacks. The name of the processes detected (powershell.exe or wmic.exe) with their respective PIDs and the extrapolation for the same in the human-readable manner, e.g. Detected encoded PowerShell command, were returned in each detection. The single instance that went undetected concerns a renamed PowerShell binary (ps.exe) one, which really points out to longer rule sets in future releases.

The memory scanner performed every complete scan less than 1 s using the normal Windows 10 virtual machine with 2 GB RAM and 2-core processor. The demand on memory was maximum ~28 MB, and CPU load was an average of less than 4%, therefore, the module is extremely lightweight and can be deployed in a legacy OT environment.

The graphical interface presented in real-time by using Flask gave non-professional users (e.g., control room operators or network technicians) the ability to study the processes without requiring direct access to console of systems or forensic tools. This online design made operations quite friendly.

Supplement of Suricata IDS Module

Our custom rule book with 12+ rules targeted the detection of fileless malware delivery and execution use cases, which increased Suricata detection function. This series of rules was aimed at:

- System.Net.WebClient Powershell Down Load Cradles
- Encoded commands(-EncodedCommand)
- Reflective loader (ReflectiveLoader, VirtualAlloc)
- mshta, rundll32 and javascript LOLBin usage
- The persistence of registry tries
- DNS requests to domains that are suspect (e.g. pastebin.com)

To obtain the detection accuracy we produced benign and malicious traffic. Suricata engine detected all the test payloads as expected and rung the bell accordingly. As indicated by examples, encoded PowerShell over HTTP was instantly noticed by the rule sid:1000002, and the exploitation of mshta and the remote script pull was captured by sid:1000008.

The true positive rate of Suricata was 95% and that of false positive was less than 5%. Administrative PowerShell scripts that technically relied on DownloadString as a means of automation were by far the biggest element of false positives. Although unpopular in OT, this demonstrates a possible requirement of context and environment-based tuning of rules.

The resource consumption was at the acceptable level. A computation of the average utilization of the CPU was about 3-5 percent and memory stress did not go beyond 40 MB. With constant packet inspection the system remained steady for more than 12 hours with no obvious reduction in performance.

Module	Detection Accuracy	False Positive Rate	CPU Usage	RAM Usage
Memory Scanner	90%	~3%	2–4%	~28 MB
Suricata IDS	95%	~4.5%	3–5%	~40 MB

Merged impact and usability

When implemented in conjunction the memory forensics and Suricata modules create a multilayered detection architecture, allowing detection of endpoint and network presents of fileless malware indicators. Several times the attacks that could not be caught on the network level (e.g., PowerShell obfuscation with HTTPS) were still detected through memory analysis. On the other hand, Suricata blocked remote LOLBin payload that did not reach the endpoint.

Moreover, the integrated dashboard enabled the analysts to match Suricata alerts with potentially suspicious processes detected in the memory, enhancing the quality of the incident responding. Through the interface, administrators were able to not only identify the threats but also to perform a few actions, such as blocking and blocking IPs and terminating processes.

6.4 Discussion

4.1 Effectiveness of Detection Modules

The high level of detection accuracy (70–95% in different scenarios) proves that the framework is efficient. Its competence in countering fileless malware, which is a loophole in the conventional signature-based techniques, which are unable to identify memory-resident threats. In comparison to the current approaches, including commercial EDR solutions (e.g., Crowd-Strike), the framework provides similar detection rates at substantially reduced resource utilization, and it is therefore more applicable in OT systems [?]. The combination of various modules increase robustness, because bundled analysis (e.g., process monitoring (These trigger memory forensics) enhances reliability of detections.

4.2 Operational Suitability

The light resource consumption of the framework (6% CPU, 115 MB memory) and low false positive rate make it suitable to be integrated with other systems. The fact that it does not interfere with OT operations, which is a crucial condition, is guaranteed by positives (010%) in systems where availability is the main concern. The validation of the framework is however limited by the fact that it depends on a simulated environment within actual OT environments, where by proprietary protocols and legacy hardware might bring about unexpected difficulties.

4.3 Limitations

There were certain issues that became obvious during testing.

A small number of false positives occurred: some SCADA applications were mistakenly flagged by the process monitoring, so those had to be looked over by hand.

• Scenario Focus: Most test scenarios worked with usual malware ways like those based on PowerShell and WMI, but advanced and changing malware was not included.

• Limited Resources: Although memory forensics is light, during the analysis, CPU usage would temporarily shoot up to 10% and this could affect the older devices.

• Live Systems Differ: Because the systems in OT are not usually put to use, they could behave differently in practice [?].

This points out how important it is to improve dynamic thresholding and include more kinds of test conditions.

Looking at how theater stories compare to literary works

The framework meets the advice offered in the literature, for instance, Baldin (2019) argues for memory forensics and Hussain (2023) asserts the need for integration among detection tools . Because it makes use of open-source tools that are suitable for OT, the framework does not need the same extensive resources as commercial systems. Unlike NIST SP 800-82 guidelines, the framework uses network segmentation and detects anomalies, which are both made specifically for fileless threats. Nevertheless, due to limited resources, NLP does not use advanced machine learning like what has been proposed by Hussain (2023).

6.5 Conclusion

In an OT environment imitation, the framework for fileless malware detection and mitigation demonstrated high precision, generated very few errors, and did not use many system resources.

Combining process monitoring, memory analysis, Suricata's methods, and quick alert generation gives a solid defense for OT systems against silent fileless attacks. Even so, there are obstacles such as false positives that can affect the results.

Even though simulation-based testing is available, the framework provides a simpler, less resource-consuming way for places with low resources. The project's goals were met by the results, and a plan for future changes is explained in Chapter 7.

CHAPTER 7

7.1 CONCLUSION

The research and development work offered in this thesis shows that the development of a tailor-made hybrid detection system is viable and that the methods work in the detection and mitigation of fileless malware in Operational Technology (OT) contexts. This project changed a lot between its conception and its actual execution, progressing away of heavyweight forensic capabilities, such as WinPMem and Volatility, into a simple, real-time Python-based application to identify the contents of volatile memory. Simultaneously, the network intrusion detection component was strengthened by writing and deploying an exhaustive set of Suricata rules explicitly developed with the purpose of scouting a wide range of fileless malware activities at multiverse levels of the network.

This aspect is also one of the main contributions of this work since it developed a practical and focused version of behavioral detection in OT environment, which is typically underprotected and challenging to protect in terms of computing resources requirements, real-time peculiarities, and its incompatibility with classical antivirus agents and endpoint detection and response solutions. The memory forensic module can operate at a relatively low cost, in that it does not have to use the full memory dump to achieve adequate detection of in-memory techniques like Invoke-Expression, -EncodedCommand, as well as WMI abuse, because of tapping into native system information through psutil. In fact, it uses minimal CPU and memory (scanning operations take less than one second and less than 30 MB of memory are required during scanning operations) - thus, the module is well inside the tolerances of the common OT hardware.

The second essential element that was Suricata with custom rule sets is very efficient in finding the indicators of fileless malware network-wide. These rules applied in this project extend past generic threat signatures and focus on encounters in the real world such as:

- Obfuscated functions and the download cradles in PowerShell commands
- Reflective DLL loading used in process injection Process injection Process injection used in process injection techniques such as reflective DLL loading Used in process injection techniques such as reflective DLL loading
- The misuse of LOLBin through binary functions in the system like mshta.exe and rundll32.exe
- Suspicious registry tenacity pointers
- DNS queries to known data exfiltration sites (e.g. Pastebin)

As found through rigorously testing, this module registered a true positive detection rate of 95 percent with few false positive readings indicating that the module is able to identify the presence of infection in the early stages of the infection before it reaches to an extent that is hard to deal with. The mitigation, which consists of real-time IP blocking via Windows Firewall and on-the-fly process termination features, further provides us a degree on being proactive on the defense side instead of passively observing what the malware does.

These two detection layers, which are memory and network, may be considered one of the most significant gains of the project; this comes in the aspect of integrating them in a single dashboard using Flask. The interface would help system administrators, including those who are not experts in cybersecurity, to visualize live threats, probe suspicious activity and respond at once. This ease of use and simplicity is in line with the industrial control network practical requirements, device uptime and the continuity of operations is of the essence.

And one more critical success factor is the extensibility of this system. The memory scanner logic and rulebase of Suricata can be extended to extend malware techniques or protocols. Additional examples, which could be added, are rules related to SMB-based lateral movement, and the engine of memory analysis could be enhanced on a small signature or anomaly detection model to detect unaccounted behaviors.

Contributions Summary:

- Creation of a light-weight process analyzer in memory that uses psutil to run processes in real time
- Installation of a Flask- Schema based web interface of the user interaction and reporting
- Development of a rules-based Suricata feature that goes after fileless malware behavior
- Combining of the two layers of detection to both be correlated and mitigated in real-time
- OT environment optimization by means of reduced resource utilization, quick response

7.2. FUTURE WORK

Even though the created framework shows extensive potential in identifying fileless malware in OT systems, it is possible to conduct some research to increase its capability, scalability, and flexibility.

1. Integration with Doing It your Way

The existing configuration monitors general network traffic (HTTP, DNS, IP), and in most OT settings, there are industrial protocols, such as Modbus, DNP3, and OPC-UA. Suricata should be added in the future to process these protocols or add functionality by custom scripting. It would help in identifying malware that manipulates protocols at a protocol level to disrupt PLCs of SCADAs.

2. Machine-Learned-Based Behavior Detection

Although the framework uses rule-based detectors that are currently deployed, there is a possibility of integrating lightweight machine learning models so the framework might be able to scan innovative and obscured behaviours. These models may process process arguments, JA3 hashes or command-line usage patterns and detect anomalies not defined by pre-configured rules. This would enhance resistance to zero-day fileless malware methodology.

3. Enhanced Mitigation Options

Currently, it is possible to manually block IP and terminate processes inside the system. Subsequent releases will make this more comprehensive with automatic quarantine, rules specific blocking or connection to SIEM platforms to centrally generate alerts. Things such as storing complete metadata of the processes, or dumping PCAPs of malicious sessions would also help with forensic response.

4. Real-World Deployment

The robustness of the system in a real scenario like the manufacturing systems, utility plants or smart buildings could also be tested in field using the actual set ups. It would also give clues on the level of noise, conformity with the industrial traffic movements and adoption usage rate by non-security workers.

5. Toolkit Development (Open- Source)

Open-source development Would be better by making the project available as an open-source modular toolkit: this would allow the community to improve the project. The availability of configurable rule packs, ease of deployment, and support of REST API would assist in making the solution scalable and associative.

These improvements would make the framework more proactive and flexible industrial security tool.

REFERENCES

[1] M. I. Hussain, “AI-Driven Threat Detection in Critical Infrastructure,” IEEE Access, vol. 11, pp. 45601–45613, 2023.

[2] P. Baldin, “The Rise of Fileless Malware,” SANS Institute, 2019.

[3] R. Langner, “Stuxnet: Dissecting a Cyberwarfare Weapon,” IEEE Security & Privacy, vol. 9, no. 3, pp. 49–51, 2011.

[4] A. Cherepanov, “Industroyer: Biggest Threat to Industrial Control Systems Since Stuxnet,” ESET Research, 2017.

[5] J. Slowik, “TRITON: The First ICS Cyber Attack Targeting Safety Systems,” Dragos Report, 2017.

[6] National Institute of Standards and Technology, “NIST SP 800-82 Rev. 2: Guide to Industrial Control Systems (ICS) Security,” 2015.

[7] International Electrotechnical Commission, “IEC 62443: Industrial communication networks – Network and system security,” 2021.

[8] MITRE ATT&CK Framework, “Enterprise Matrix,” https://attack.mitre.org, Accessed May 2025.

[9] T. H. Kim et al., “Fileless Malware Attack Techniques and Detection,” Electronics, vol. 9, no. 11, p. 1777, 2020.

[10] Y. Lu et al., “Memory Forensics for Fileless Malware Detection,” Journal of Cybersecurity, vol. 7, no. 1, 2021.

[11] A. Vardhan and R. Kumar, “Cybersecurity Challenges in Operational Technology,” IJCSIT, vol. 15, no. 3, pp. 23–30, 2023.

[12] J. Ullrich, “Suricata and Emerging Threat Detection,” SANS Reading Room, 2020.

[13] WinPMem, “Windows Physical Memory Acquisition Tool,” Rekall Project, https://github.com/Velocidex/WinPmem

[14] Volatility Foundation, “Volatility Framework: Memory Forensics,” https://www.volatilityfoundation.org

[15] Open Information Security Foundation, “Suricata IDS/IPS/NSM Engine,” https://suricata.io

[16] A. M. Azab et al., “BE-PAR: Behavior-Based Policy Enforcement for Android Runtime,” USENIX Security, pp. 111–126, 2016.

[17] Microsoft, “Windows Management Instrumentation (WMI),” https://docs.microsoft.com

[18] Microsoft, “PowerShell Security Best Practices,” Microsoft Docs, 2022.

[19] Y. Liu et al., “Detecting Anomalous Activities in ICS Using Machine Learning,” IEEE Trans. on Industrial Informatics, 2022.

[20] Kaspersky, “Living-off-the-land Attacks: Fileless Malware,” 2021.

[21] Cisco, “Cisco SecureX Threat Response,” Cisco Whitepaper, 2021.

[22] CrowdStrike, “Falcon Insight EDR Overview,” 2022.

[23] McAfee, “The Evolution of Fileless Malware,” McAfee Labs, 2020.

[24] Symantec, “Living Off the Land and Fileless Attack Techniques,” Symantec Threat Report, 2021.

[25] FireEye, “The Rise of Fileless Malware,” Threat Research Blog, 2020.

[26] IBM X-Force, “Fileless Malware Attacks in OT Networks,” 2023.

[27] N. Ye et al., “Anomaly Detection Techniques in OT Systems,” IEEE Trans. on Reliability, vol. 69, no. 2, pp. 794–808, 2020.

[28] Z. Durumeric et al., “Security Implications of Industrial Control Protocols,” NDSS, 2018.

[29] P. Porras et al., “Securing SCADA Systems: Challenges and Solutions,” USENIX HotSec, 2021.

[30] J. Hong et al., “SCADA Security in Power Systems,” Energies, vol. 13, no. 3, p. 653, 2020.

[31] OTbase, “Legacy OT System Integration Challenges,” Industrial Security Journal, 2022.

[32] ICS-CERT, “Analysis of Triton Malware Targeting SIS,” Department of Homeland Security, 2018.

[33] R. Villafana, “Threat Detection Using Behavioral AI,” DarkReading, 2023.

[34] M. Almgren et al., “Scalable Intrusion Detection for ICS,” IEEE Trans. on Dependable and Secure Computing, 2021.

[35] T. Holz et al., “Measuring and Detecting Malware Behavior,” IEEE TDSC, vol. 12, no. 5, pp. 542–555, 2015.

[36] P. Chen et al., “A Survey of Command and Control Techniques in Malware,” IEEE Communications Surveys & Tutorials, 2017.

[37] K. Scarfone, “Guide to Malware Incident Prevention and Handling,” NIST SP 800-83 Rev. 1, 2013.

[38] S. Zargar et al., “A Survey of Intrusion Detection Systems in SCADA Networks,” Journal of Network and Computer Applications, 2017.

[39] G. Loukas, Cyber-Physical Attacks: A Growing Invisible Threat, Butterworth-Heinemann, 2015.

[40] J. W. Smith, “Secure Network Design for ICS,” IEEE Security & Privacy, 2022.

[41] K. Salah et al., “Machine Learning for Threat Detection in ICS,” IEEE Access, 2021.

[42] C. Tankard, “Advanced Persistent Threats and Fileless Malware,” Network Security Journal, 2019.

[43] B. Johnson, “Detection Challenges in SCADA Systems,” InfoSec Journal, 2021.

[44] S. Mohammadi et al., “Cybersecurity for Critical Infrastructure: OT Perspective,” Computers & Security, vol. 89, 2020.

[45] R. Anderson, Security Engineering, 3rd ed., Wiley, 2020.

[46] Z. Tari, “Security and Privacy in Smart Grids,” IEEE TSC, vol. 9, no. 3, 2022.

[47] L. Spitzner, Honeypots: Tracking Hackers, Addison-Wesley, 2002.

[48] J. Meseguer, “Formal Methods in Industrial Security,” IEEE Industrial Informatics, 2020.

[49] T. Toth and C. Kruegel, “Accurate Buffer Overflow Detection via Abstract Payload Execution,” RAID, 2002.

[50] ICS Security Working Group, “Top 20 Secure PLC Coding Practices,” SANS Institute, 2021.