Term Paper II

CS 6301.002 Special Topics in Computer Science- Spring 2023

Critique by: AKASH RAJUBHAI RAMANI

Net Id: AXR200012

Title of Paper: Securing Remote Access Networks Using Malware Detection Tools for Industrial Control Systems

Author of Paper: Okechukwu Ude, Bobby Swar

Publication: IEEE Network

Abstract:

The utilization of deep learning presents a technique to get an information from the sensors. Especially data from IoT devices operating. On the other side the multiple layered structure for deep learning is suitable for application and programming in edge. This paper proposes an approach that incorporates deep learning into IoTs for use in the computing environment for edge. Thereby bridging the gap between these two technologies to improve the overall throughput of the system. Given very limited processing capacity, how to we achieve complex detection tasks. Various deep learning methods are compared to leverage the most suitable algorithms suitable for the task at hand.

A performance assessment is conducted to examine the execution of many tasks pertaining deep learning in an edge computing scenario using our approach. The results reveal that our method surpasses other strategies in terms of performance. The offloading strategy proposed in the study is intended to reduce the burden on edge nodes by transferring part of the deep learning workload to cloud servers. The study shows that this approach significantly improves the overall performance in our targeted tasks.

In conclusion, the proposed approach provides an efficient solution for incorporating deep learning into IoT devices operating in edge computing environments. The use of CNN and LSTM networks optimized and trained on cloud servers and deployed to edge devices for inference, along with the offloading strategy, provides a high-performance solution for complex detection tasks. This approach has the potential to improve the overall throughput of the system, making it a promising solution for real-world applications in various industries.

Introduction:

Industrial Control Systems (ICS) - It refers to the hardware and software-based technologies used to tracking and regulating the physical processes within a set of industrial environments, such as manufacturing plants, power stations, and water treatment facilities. These systems include a variety of devices, such as programmable logic controllers (PLCs), some sensors, and the human-machine interfaces (HMIs), that work together to manage and automate processes. ICS play a very important role in order to maintain the efficiency, safety, and reliability of industrial operations by regulating the flow of materials, energy, and information. They also provide operators with real-time data and alerts to enable timely decision-making and response to potential problems.

One example of the use of Industrial Control Systems (ICS) is in a manufacturing plant that produces automobiles. The ICS would be used to control and monitor various processes such as the assembly line, painting process, and quality control.

For instance, the programmable logic controllers (PLCs) within the ICS would be responsible for controlling the movement of the assembly line and the robots that perform welding and painting operations. The sensors would monitor the quality of the paint job and detect any

defects, while the HMIs would provide the plant operators with real-time data on the status of the production line and any issues that arise.

Overall, the ICS would help to ensure that the manufacturing process is efficient, safe, and produces high-quality automobiles by regulating the various processes involved in production.

Malware detection tools for ICS - It refers to software programs designed to identify and prevent malicious software from infecting and disrupting critical industrial processes. These tools use a range of techniques such as Signature-based detection, behavior-based detection, and machine learning are three common approaches used in cybersecurity to detect and prevent threats. Signature-based detection involves comparing the characteristics of incoming traffic or files against a database of known signatures of malicious software. Behavior-based detection focuses on analyzing the behavior of software, looking for suspicious patterns that may indicate a threat. Machine learning involves training algorithms to recognize patterns in data, allowing them to identify and classify new threats based on similarities to known threats. Each of these approaches has its strengths and weaknesses and may be used in combination to create a more comprehensive security solution. This is to detect and respond to potential malware threats within ICS networks.

Industrial Control Systems (ICS) serve as a crucial component of national infrastructure for industrial development. While the integration of Information Systems (IS) with ICS has led to significant advancements in critical infrastructure sectors, such as controlled-environment agriculture, automated train systems, and smart homes, it has also given rise to new vulnerabilities, such as Remote Access Trojans (RATs). Conventional RAT detection methods rely on monitoring network traffic or scrutinizing event logs on host systems. This research seeks to identify and address RATs by comparing actual system utilization to reported utilization, conducting a GAP-analysis of open-source RAT detection methods to pinpoint inadequacies, and integrating control algorithms into the source code for a comprehensive solution.

An innovative study aims to devise an instrument that discerns and eradicates Remote Access Trojans (RATs) from the intricate network of Industrial Control Systems (ICS). The RAT detection mechanism shall hinge on the versatile ClamAV open-source antivirus engine and be executed with the elegance of the Python programming language, bolstered by the Yara-python library to promote secure coding practices. This research stands apart from the conventional antivirus software that merely cross-references applications against pre-existing databases of known hazards. Instead, it scrutinizes the behavioral blueprints of active applications and processes to identify suspicious anomalies that warrant further examination.

Their Literature Review:

Knowles et al. highlight the low priority given to the security of ICS its reliability on "security through obscurity" as a safeguard against attacks. However, with the new technologies in open communication in networked ICS, this approach is no longer feasible. The prioritization of

availability over confidentiality is another challenge in securing ICS, leading to potentially counterproductive security controls.

Here's a rephrased and summarized version of the given text:

1. Security controls prioritization - The CIA and AIC, each dimension of the CIA triad - confidentiality, integrity, and availability - is of equal importance in ensuring the security of information and information systems, but ICS security controls are often implemented prioritizing availability over confidentiality, leading to inefficient and ineffective controls.
2. Convergence of IT and OT - The integration of IT and OT resulted in an upsurge of security vulnerabilities, exposing them to more threats.
3. Communication protocols - The absence of access control policies like authentication in ICS communication protocols such as MODBUS during their design can expose them to threats from attackers who can create fake identities and send malicious commands.
4. Malware - The Stuxnet attack on the centrifuges in Iran alerted the world to the need for advanced security in ICS. As a result, various examples of malware have emerged in the ICS landscape, designed to steal information, disrupt services, and cause chaos in industrial control systems.

Remote Access Trojans:

A Trojan, also known as a Trojan horse, is a form of malicious software that camouflages itself as a genuine application or program, but once installed on a computer, can allow unauthorized access or cause damage to the system.

A remote Trojan is a type of Trojan malware network without the victim's knowledge or consent. Trojans are often distributed through email attachments, software downloads, or infected websites, and can perform a variety of harmful actions, including stealing sensitive data, installing additional malware, and hijacking the victim's computer for use in a botnet. Trojans are a common threat to computer security and can cause significant damage to individuals and organizations if left undetected and unremoved on system from a remote location.

Working of Remote Access Trojan:

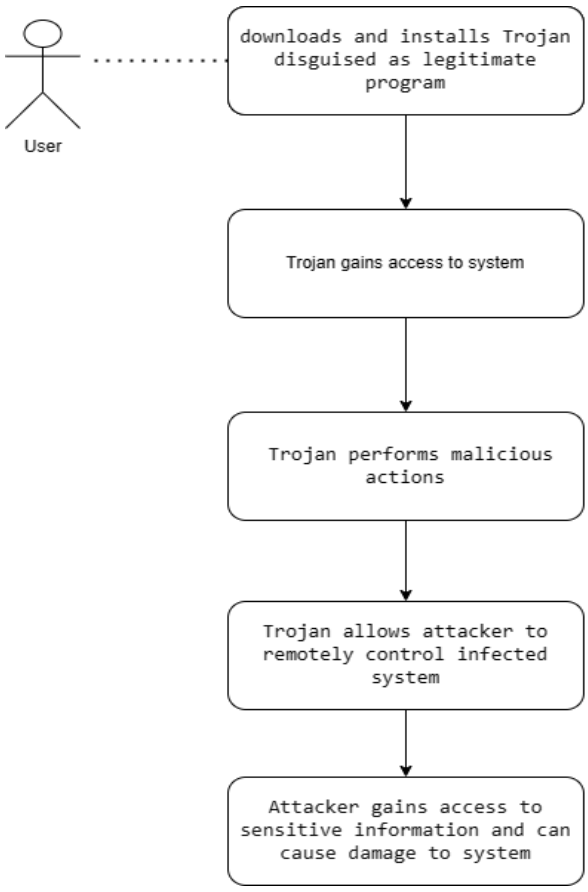

Figure 1. Working of Remote Access Trojan

Approaches and Methodology:

The objective of this study is to create a tool that can identify and remove the Remote Access Trojans (RATs) in Industrial Control Systems (ICS). This tool is host-based, utilizing the open-source ClamAV antivirus engine and is programmed using Python. To define malware, the tool is integrated with the Yara-python library, which is also utilized by VirusTotal.

In this study, for open-source, its limitations of existing malware detection tools are examined and resolved by implementing a host-based tool. The implementation follows the principles of modular software development, with high cohesion and low coupling.

This research aims to improve the process on detecting and eliminating of Remote Access Trojans (RATs) in industrial control systems by developing a new host-based tool. The tool is designed using the tight interconnectedness and loose coupling principle to facilitate the development of software in a modular fashion and incorporates features that address the weaknesses of existing tools. The study investigates the effectiveness of previously developed methods for RAT detection in ICS and identifies the key features that should be incorporated within a more recent instrument to surmount the constraints of prior techniques.

Algorithms were developed and integrated into a new host-based tool for RAT detection in industrial control systems. The detected RAT detection utilities were analyzed and evaluated based on their capacity to function on the given host based system with a minimum detection frequency of 80%. The effectiveness and efficiency of each approach were then deduced by inspecting the detection frequency of trial RAT samples utilized in the inquiry.

To establish the shortcomings of each tool, a GAP based approach for analysis was implemented. After identifying the gaps, control algorithms were created and integrated into a novel host-based tool for RAT detection in ICS.

The subsequent descriptions present a condensed depiction of the source code algorithm that was devised to address the inadequacies of the formerly accessible open-source RAT detection software:

1. Retrieve a comprehensive inventory of all applications currently installed on the system.
2. Retrieve a comprehensive list of all active processes and applications currently running on the system.
3. Assess and calculate the respective proportion of system resources, including CPU, memory, disk, and network, being utilized by each individual process and application.
4. Export the obtained calculations to a spreadsheet or similar data storage format.
5. Compare the anticipated system resource usage, which is preconfigured by the system administrator, with the actual usage calculated in Step 3.

6. In case of unexpected values, proceed to sub-step 6a. If there are no unexpected values, continue to Step 7.
    a. Get a list of all the unknown or unexpected applications and processes that are needed for the hash value calculation.
    b. After the computations, a comparison for the hash values is done to get system administrator's database.
    c. If there is a match, kill the application or process, make sure to log that it is detected, and then alert the system administrator.
    d. Else: kill the application or process, make sure to log that it is detected, and then for more analysis, alert the system administrator
7. Resume to the idle state until the next scanning operation.

This algorithm is designed to fetch information about installed and running applications, calculate system component usage, and compare it to pre-configured values to identify unexpected applications and processes. It utilizes hash value calculation to determine whether the unknown applications or processes are RATs or not. If the match is found, the application or process is terminated, and the system administrator is notified. If no match is found, the file is submitted to the application called the VirusTotal and there the preceding analysis are carried out and based on the outcome they decide whether to alert the system administrator or not. The algorithm operates in standby mode between scans to conserve system resources.

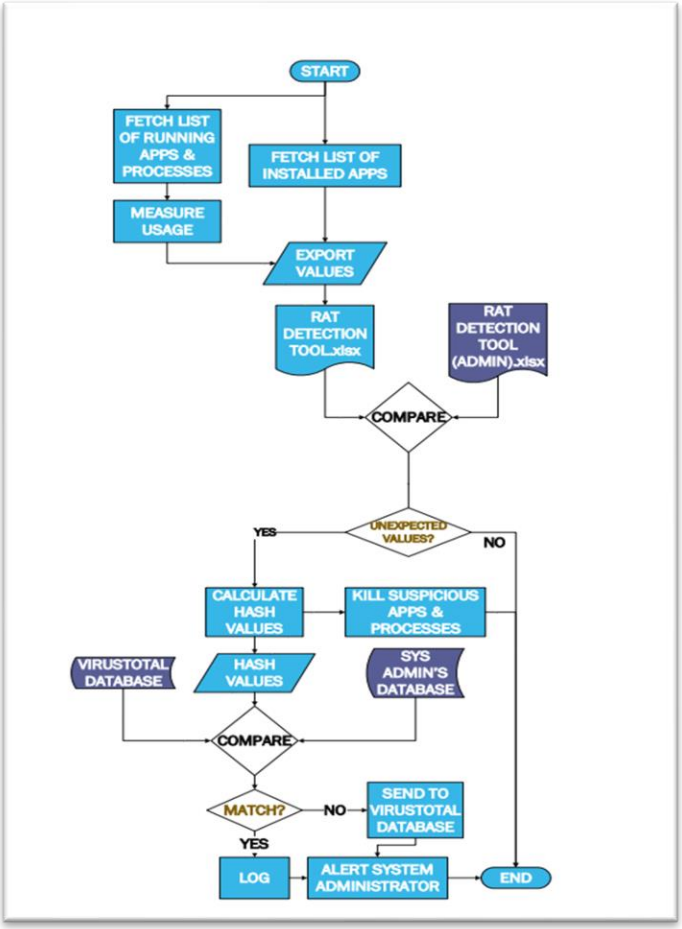

Figure 2 RAT Detection Tool Flowchart

Evaluation and Analysis:

By subjecting the identified RAT detection tools to scrutiny, a comprehensive gap analysis was conducted to establish the current capabilities of each tool in the realm of host-based RAT detection. The analysis also sought to establish the ideal state of capabilities for each tool and, consequently, identify the extent of the shortfall between the current and desired capabilities of each tool.

To evaluate the effectiveness of the newly developed RAT detection tool,

They set up an environment for testing purposes, and that was done using VMware's workstation Pro Application. And subsequently the VMWares were created for both of the operating systems, i.e: The Windows and Linux OS.

And within this particular test environment, subsequent folder named "RM" was added in the user's directory. This "RM" folder contained a subfolder called "RAT Detection Tool". Additionally, a subfolder named "sheets" was created within the folder named "RAT Detection Tool" folder. The "sheets" folder contained several files such as "System Administrator sheets", which contained two pre-configured files (one xlsx and another csv). They used these files do analyze and to compare the outcomes generated by the tool. The "sheets" folder also contained other files too.

Filenames, in-fact are quite inadequate to differentiate weather or not one is an allowed remote remote administration tool or a spurious RATs. Therefore, the hash calculation feature of the tool was found to be essential for more accurate detection.

To test the hash calculation feature, the source code hardcoded the hash values for a pre-decided malicious access sample and an allowed sample for our Remcos's Remote Administration Tool stub. What the tool did was it compared the hash-values of the samples to the VirusTotal database. This study summarizes the results of the hash scans performed on both samples.

Drawing from the findings presented, one may deduce that the source code developed for the RAT detection tool successfully identified remote access stubs and differentiated between legitimate remote administration tools and RATs. The comparison scan results demonstrated the limitations of using filenames as a means of distinguishing between the two, while the hash scan results showed the effectiveness of hash-checking in identifying RATs.

Furthermore, the source code provides an automated and systematic approach to detecting RATs and ensuring the integrity of the system. The tool's ability to export data to spreadsheets and compare it to pre-configured system usage data enables easy identification of unexpected values and unknown/unexpected applications and processes.

However, the limitations of the The researcher utilized developer account on VirusTotal are acknowledged, and it is recommended that further testing be carried out with a full VirusTotal account to maximize the tool's effectiveness.

Limitations:

Conducting studies can present various challenges and limitations for researchers, as noted in the analysis provided. The inadequacy of a genuine physical setting for assessing the efficacy of the tool is a significant obstacle, as the ICS threat landscape is highly volatile, necessitating the development of detection filters based on cognitive learning. Moreover, exclusive constraints associated with ICS providers and industries can pose compatibility issues, thereby further

restricting research efforts. Another impediment is the absence of freely available authentic samples of malicious RATs, which impedes the analysis of malware patterns and the creation of Yara-python definitions for various malware families. However, the research offers significant insights and develops a RAT detection tool that can improve detection capabilities despite these challenges and limitations.

Conclusion and Future Scope:

The idea of scheduling for efficiently splitting up the neural network is novel, and the results look promising to a great degree. The scheduling algorithms aims to reducing the latency of the network thereby increasing the overall throughput of the system. Deep leaning, though achieves staggering accuracy and precision, it is still and ongoing field of research/ engineering. There is always a scope for improvement by constantly using different feature extraction methodologies, altering the hyper parameters to improve on accuracy, use of heuristic functions that would help us to get close to global optimal solution, reduce the running time of networks.

Talk about how the network would plan on to communicate the results back to the server and furthermore, if those data need not be stored, can it be possible to device an action plan in edge devices itself that improves the response time for Iot devices.

References:

[1] Okechukwu Ude; Bobby Swar. Securing Remote Access Networks Using Malware Detection Tools for Industrial Control Systems
[2] Trend Micro Incorporated, "Industrial Control System," 9 June 2020.
[3] F. Imam, "Malware spotlight: What is a Remote Access Trojan (RAT)?," 3 December 2019.
[4] D.-Y. Kim, "Cyber Security Issues Imposed on Nuclear power plants,".
[5] A. A. Awad, S. G. Sayed and S. A. Salem, "Collaborative Framework for Early Detection of RAT-Bots Attacks", IEEE Access,

Securing Remote Access Networks Using Malware Detection Tools for Industrial Control Systems

Okechukwu Ude, Bobby Swar
Information Systems Security and Assurance Management
Concordia University of Edmonton,
Alberta, Canada
oude@student.concordia.ab.ca, bobby.swar@concordia.ab.ca

***Abstract*—With their role as an integral part of its infrastructure, Industrial Control Systems (ICS) are a vital part of every nation's industrial development drive. Despite several significant advancements - such as controlled-environment agriculture, automated train systems, and smart homes, achieved in critical infrastructure sectors through the integration of Information Systems (IS) and remote capabilities with ICS, the fact remains that these advancements have introduced vulnerabilities that were previously either non-existent or negligible, one being Remote Access Trojans (RATs). Present RAT detection methods either focus on monitoring network traffic or studying event logs on host systems. This research's objective is the detection of RATs by comparing actual utilized system capacity to reported utilized system capacity. To achieve the research objective, open-source RAT detection methods were identified and analyzed, a GAP-analysis approach was used to identify the deficiencies of each method, after which control algorithms were developed into source code for the solution.**

Keywords—Industrial control systems, operational technology, remote access trojans, malware detection, Purdue control hierarchy, hash calculation, confidentiality, integrity, availability

I. INTRODUCTION

The term "Industrial Control Systems" is used to refer to any combination of devices, systems, networks, and controls with the purpose of operating or automating industrial processes [1].

Over the years, critical infrastructure' control systems pivotal to the adequate functioning of sectors like the oil & gas, manufacturing, water, and power industries have been repeatedly lined up in the crosshairs of cyber-attackers. With the main infection channel being the internet via phishing emails, the threats introduced include crypto-mining, ransomware, remote access trojans (RATs), and a host of others [2].

RATs are a rampant threat that has become synonymous with major cyberattacks. Initially, RATs referred to Remote Access Technologies, but over time have become more commonly used to refer to Remote Access Trojans - a malicious variant of remote access technologies [3]. RATs are a form of malware that grants an attacker administrative access to a remote device, facilitating covert surveillance together with unfettered and unauthorized access, thereby establishing the attacker's foothold in a target system or network [4].

Presently, adequately patched and regularly updated operating systems, browsers, and antivirus software, as well as other physical and logical controls, work at protecting the host systems while IDS/IPS work at monitoring the network and its hosts for RAT activity. As robust as protective measures are, RATs have still found their way into systems and networks and managed to remain hidden.

Most cyberattacks on ICS are launched using RATs such as Stuxnet [5]. According to a Department of Homeland Security report, at least 55% of the 245 reported ICS attack cases in 2015 were attributed to RATs [6]. Kaspersky's "The State of Industrial Cybersecurity" report, states that 68% & 66% of companies see Targeted Attacks (via RATs) & Conventional Malware respectively as major concerns for their control systems [7]. This statistic underlines the need to increase the detection strength of malware-detection implementations.

This research aims to develop a tool for the detection and elimination of RATs in ICS. The RAT detection tool will be based on the ClamAV open source antivirus engine. The implementation was encoded in the python programming language and will utilize the Yara-python library in line with cybersecurity best practices. This research - unlike antivirus software which compares applications against databases of known threats, analyzes process patterns of running applications and processes, then flags suspicious applications or processes for further analysis.

II. LITERATURE REVIEW

A. Security Issues in ICS

Knowles et al. [8] explain that giving the security of ICS a low priority has been the default stance of ICS stakeholders while depending on "security through obscurity" to safeguard ICS from attacks. Classifying ICS as monolithic (using minicomputers), distributed (geographically distributed computing), networked (process control networks), and web-based, Knowles et al. [8] further explained that although "security through obscurity" in monolithic and distributed ICS may have worked, from networked ICS onwards the increase in attack susceptibility due to the introduction of open communication technologies has discouraged the use of obscurity to ensure security.

Some other challenges faced in securing ICS include:

1) *CIA vs. AIC:* The dimensions of the CIA triad are conceptually equally important. According to Knowles et al. [8], some aspects of the CIA triad are deemed more expedient than others for reasons which could be economical or strategic. In the area of information security, such a perception can be seen in the implementation of controls according to CIA - confidentiality being the paramount goal; while in the area of ICS controls are implemented according to AIC - availability being the paramount goal. The reversal of security control's prioritization is a major reason behind inefficient, ineffective, and potentially counterproductive ICS security controls [8].

2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS) | 978-1-7281-6207-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPS49255.2021.9468212

2) *The convergence of IT & OT:* Idrissi et al. [9] report that the evolution of ICS i.e. the integration of I.T and O.T, has led to the increased exposure of ICS to threats and as a result has led to an increase in the number of vulnerabilities in ICS. Idrissi et al. [9] map the rate of increase in vulnerabilities in Table I:

TABLE I. NUMBER OF VULNERABILITIES FROM 2015 TO 2018 [9]

Year	2015	2016	2017	2018
Number of Vulnerabilities	182	187	322	415

Some other security concerns include breaks in business continuity due to catastrophic events, network downtimes that affect customer-facing systems, and the inability to identify, measure, and track organizational risk.

3) *Communication protocols in IT & OT:* Idrissi et al. [9], Babu et al. [10], and Fan et al. [11] report that the traditionally isolated nature of ICS is the reason why ICS communication protocols do not take access control policies like authentication into account in their design. The lack of access control measures on ICS communication protocols (like MODBUS) enables attackers who can create counterfeit identities to access the network and send erroneous messages as well as malicious commands to ICS components. Babu et al. [10] add that the failure to integrate access control via authentication into the ICS design does not mean that authentication and encryption cannot be used with ICS systems.

4) *Malware:* Fan et al. [11] describe the Stuxnet malware-based attack on Iranian centrifuges as a worldwide wake-up call to the need for advanced security in ICS.

Following in Stuxnet's wake, the likes of Duqu, Flame, Gauss, and Shamoon are just a few of the malware examples that have since plagued the ICS landscape, and just like Stuxnet, each malware was designed to thieve information, deny services, and cause all-out mayhem in industrial control systems [12]. Kim [12] reports that following the events of the Stuxnet attack in 2010, the source code for the Stuxnet malware was posted online, serving as a prototype for future generations of cyber-attacks.

B. Remote Access Trojans

In securing industrial control systems, remote access to the control systems is a necessary functionality that facilitates quick troubleshooting and easy configuration irrespective of geographic location [13]. RATs are a variant of trojans - a type of malware, which exploits the need for remote access functionalities by ICS [3]. RATs are a form of malware that grants an attacker administrative access to a remote device, allowing covert surveillance, together with unfettered and unauthorized access, thereby establishing the attacker's foothold in a target system or network [4].

According to research, one of such cyberattack campaigns was deployed via a phishing email containing a pdf document offering safety measures against the coronavirus. The document contained executables for a Remcos RAT dropper which would run alongside a VBS file, thereby executing the malware [14]. Gatlan [15] explains that efforts at gaining persistence on the target device are made by the Remcos RAT through attempts to affix a Startup Registry key at "HKCU\Software\Microsoft\Windows\CurrentVersion\Run Once". If successful, the established persistence allows the Remcos RAT to restart whenever the computer is restarted. Immediately after downloading every needed extension, the RAT begins to log the target user's keystrokes, which are stored in a log.dat file in a temporary local or OneDrive folder. Fig. 1 shows an illustration of the Remcos attack process:

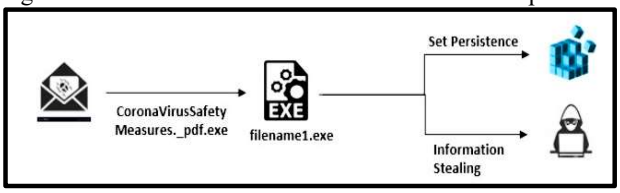

Fig. 1. Remcos RAT Attack Chain [16]

The stored keystrokes are then exported to a "command & control" server hosted at a predetermined IP address.

The practice of stealthy, ongoing hacking directed at the accumulation of data over time, rather than causing damage to information or systems, is identified as an Advanced Persistent Threat (APT) attack, an attack-type where the power of RATs is greatly utilized. RATs are ideal for APTs because not only do RATs not reduce a computer's performance or automatically delete files upon installation but also because RATs are quite adaptable - just like live rats [3].

Efforts have been made at creating tools & methods for the detection of these RATs and are identified in Table II and III:

TABLE II. NETWORK-BASED RAT DETECTION METHODS

Researchers	Network-Based Methods
Jiang & Omote (2015) [17]	Detecting RATs in the early communication stage by depending on the features of network activity that can be derived from the TCP header.
Pallaprolu et al. (2016) [18]	An ensemble-based label propagation technique that detects RATs in huge, processed, and unlabeled datasets using a big data engine.
Wu et al. (2017) [19]	A system for detecting RATs in networks by extracting inter-arrival time sequences, payload-sized packet sequences, and IP traffic route packet sequences, then analyzing the flow slices from the inter-arrival time sequences.
Mimura et al. (2017) [20]	By analyzing proxy server logs and observing network traffic patterns like interval sizes, precision, recall, and F-measures, RAT activities were found to have characteristic behaviors.
Zhu et al. (2019) [21]	Six unsupervised classification algorithms are implemented on a network traffic dataset based on the states of four uncorrelated network activity features obtained from TCP sessions to identify and illustrate the difference between RAT-traffic and legitimate sessions.

TABLE III. HOST-BASED RAT DETECTION METHODS

Researchers	Host-Based Methods
Mimura et al. (2017) [22]	A brute-forcing tool is developed and used to overcome obfuscation while disinfecting a RAT-infested document file.
Awad et al. (2019) [23]	A framework was proposed consisting of two agents-- the host agent and the network agent, responsible for monitoring the host's behavior and the network's traffic respectively for malicious patterns.

Noticeably, little work has been done in the area of RAT detection on host systems compared to the work done in the area of RAT-detection on networks.

III. METHODOLOGY

This research develops a host-based tool for the detection and elimination of RATs in ICS referencing the ClamAV open

source antivirus engine, is encoded in the python programming language and is configured to utilize the Yara-python library for malware definition as is also used by VirusTotal [24].

This research identifies and addresses some of the weaknesses in the detection capacities of present open-source malware-detection tools used for ICS. This implementation follows the high cohesion & low coupling recommendation for modular software development [25]. The research addresses the following questions:

- What effective and efficient methods were formerly developed to address the problem of RATs in industrial control systems?
- What features need to be integrated into a newer tool to eliminate the shortcomings of earlier methods?

Some samples of available open-source RAT detection tools have been identified and analyzed identified based on their capability to work on a host system at a detection rate of 80% or more. The effectiveness and efficiency of identified methods were deduced based on the tool's detection rates of test RAT samples used in the research.

A GAP-analysis approach was used to identify the deficiencies of each tool, after which control algorithms were developed into source code for the solution.

A. Source Code Algorithm

The statements below are a simplified representation of the algorithm for the source code based on the identified gaps:

1) Fetch a list of all installed applications.
2) Fetch a list of all running applications & processes.
3) Measure the portion of each system component (CPU, memory, disk, network) being consumed by each application & process.
4) Export values to a spreadsheet.
5) Compare expected system usage (contained in the system administrator's pre-configured spreadsheet) to calculated system usage (as compiled in Step 4).
6) If there are unexpected values in step 5, proceed to sub-step 6a; else proceed to Step 7.
    a) Compile a list of unknown/unexpected applications & processes for hash value calculation.
    b) After calculation, compare the values to the system administrator's database and the VirusTotal database.
    c) If there is a match, kill the app(s) and/or process(es), log the detection and alert the system administrator.
    d) If there is no match, kill the app(s) and/or process(es) log the detection, submit the file to VirusTotal for further analysis, and alert the system administrator.
7) Return to standby mode until the next scan.

The flowchart for the algorithm is depicted in Fig. 2 while some sections of the source code for the algorithm are illustrated in Fig. 3 and 4:

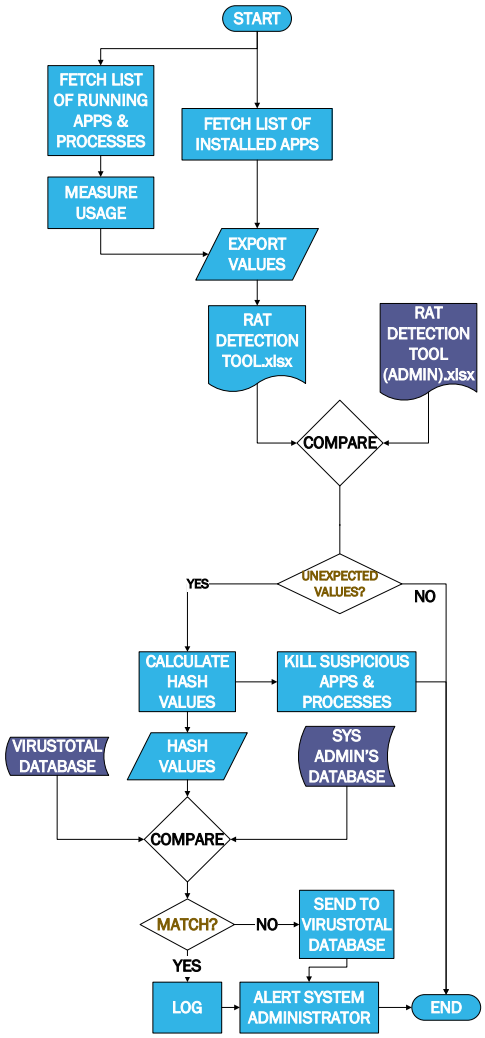

Fig. 2. RAT Detection Tool Flowchart

```
# FETCH A LIST OF ALL INSTALLED APPLICATIONS
while IA_counter == 0:
    def foo(hive, flag):
        a_reg = winreg.ConnectRegistry(None, hive)
        a_key = winreg.OpenKey(a_reg, r"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall",
                               0, winreg.KEY_READ | flag)

        count_subkey = winreg.QueryInfoKey(a_key)[0]

        application_list = []

        for i in range(count_subkey):
            application = {}
            try:
                asubkey_name = winreg.EnumKey(a_key, i)
                asubkey = winreg.OpenKey(a_key, asubkey_name)
                application['name'] = winreg.QueryValueEx(asubkey, "DisplayName")[0]

                try:
                    application['version'] = winreg.QueryValueEx(asubkey, "DisplayVersion")[0]
                except EnvironmentError:
                    application['version'] = 'undefined'
```

Fig. 3. Fetching the List of Installed Applications (Step 1)

```
# GATHER INFORMATION ON ALL PROCESSES
while APM_counter == 0:
    # fetch information on running processes
    def fetch_process_info():
        procs = []  # this list will contain all process dictionaries
        # fetch all process_info at once
        for process in psutil.process_iter():
            with process.oneshot():
                pid = process.pid  # fetch process id
                if pid == 0:  # the "System Idle Process" for Windows NT is being exclud
                    continue
                name = process.name()  # this will get the name of the process executed

                # TO GET THE TIME THE PROCESS WAS FIRST GENERATED
                try:
                    create_time = datetime.fromtimestamp(process.create_time())
                except OSError:  # for system processes using boot time instead
                    create_time = datetime.fromtimestamp(psutil.boot_time())
```

Fig. 4. Fetch Information on Running Applications & Processes (Step 2)

IV. RESULTS

A. Evaluation & Analysis

Using the RAT detection tools identified as subjects, a gap analysis was carried out which identified the present state of each tool's capabilities in host-based RAT detection, the desired state for each tool's capabilities, and the gap between the present state and desired state for each tool.

The gap analysis is illustrated in Table IV:

TABLE IV. A GAP ANALYSIS OF SELECTED OPEN-SOURCE RAT-DETECTION TOOLS

S/ No	Tool	Present State	Desired State	Gap
1	AIDE (Advanced Intrusion Detection Environment)	Supports only Unix-based operating systems	Host-based process-integrity checker with cross-platform compatibility	Non-emulated cross-compatibility
2	Samhain	Host-based process-integrity checker. Works on Windows only through the *cygwin* emulator.	Host-based process-integrity checker with cross-platform compatibility	Non-emulated cross-compatibility
3	Snort	Network-based detection tool	Host-based process-integrity checker with cross-platform compatibility	Host-based detection.
4	OSSEC Host Intrusion Detection System	Works on only Unix-based operating systems	Host-based process-integrity checker with cross-platform compatibility	Non-emulated cross-compatibility
5	Sagan	Only supported on Linux distributions	Host-based process-integrity checker with cross-platform compatibility	Non-emulated cross-compatibility
6	Security Onion	Standalone Linux distribution	Host-based process-integrity checker with cross-platform compatibility	Non-emulated cross-compatibility

The "Present State" column identifies the present capabilities of each tool. The "Desired State" column states the functionality that the research aims to integrate into the source code for the RAT detection tool. The "Gap" column identifies the capabilities that are desired but not present in each tool. The combination of the contents of the "Desired State" and "Gap" columns produces a summarized description of the developed RAT detection tool.

B. Simulation & Testing

To experiment on the effectiveness of the new tool, a test environment was set up using Vmware Workstation Pro and virtual machines were created for both Windows & Linux operating systems.

For the test environment, a folder was created within C:\Users containing the following:

1) A folder called "RM", with a "RAT Detection Tool" folder inside.

2) A folder called "sheets" inside the "RAT Detection Tool" folder. This folder contains the following:

a) A folder named "System Administrator sheets" containing two preconfigured "RAT Detection Tool (Admin).xlsx" and "Processes-Hash List.csv" files used in a comparative analysis of the tool's results.

b) List of Undesirables.xlsx

c) RAT Detection Tool.xlsx

d) Unknown Processes.csv

Default applications & processes were left installed and running after the initialization of each test virtual machine. For this research 70 RAT samples from 50 different families were collected and used for the test. The results of the comparison between the system's present state and the system administrator's preconfigured state are as shown in Table V:

TABLE V. RESULTS OF THE COMPARISON SCAN ON TESTED SAMPLE'S STUBS

S/ No	Family Name	Language	Debut Year	Comparison Scan Result
1	Alusinus	Delphi	2013	Detected (Server.exe)
2	Babylon	C++	2015	Detected (Server.exe)
3	BackConnect	.NET	2014	Detected (Server.exe)
4	Bozok	Delphi	2012	Detected (Server.exe)
5	BXRAT	.NET	2014	Detected (Server.exe)

The table has been truncated for the sake of brevity.

The results show that each remote access stub that was run as part of the test was identified each time as an application named "Server.exe". The results show that filenames are not sufficient for use in distinguishing stubs for legitimate remote administration tools from stubs for RATs. This insufficiency highlights the need for the hash-calculating aspect of this research.

For this research, the hash values of a malicious sample and a legitimate sample of a Remcos Remote Administration Tool stub were hardcoded into the source code and compared to the VirusTotal database. The choice of the RAT family (Remcos) used for this test is based on the availability of samples and the rate of occurrence of this RAT. The results of the hash scans carried out on both samples are summarized and illustrated in Table VI and Fig. 7 & 8:

TABLE VI. RESULTS OF THE HASH CHECK ON TESTED SAMPLES [26]

S/ No	Family Name	Comparison Scan	Hash Value (SHA1)	Hash Check Result	
				Sys Admin	*Virustotal*
1	Remcos RAT (legitimate)	Detected (remcos_agent.exe)	0e5b8b6ce7b39fff288a6b89d501e49cfb1b52f9	0	0
2	Remcos RAT (malicious)	Detected (remcos.exe)	59b07235c43bc3098a2bb5ef05fc8c8d0484499c.[a]	1	1

[a] Hash value hardcoded into sample [26].

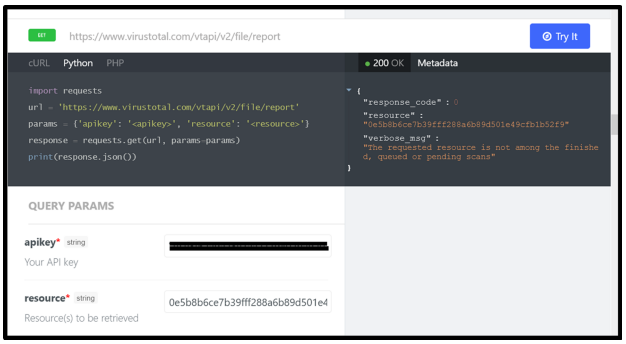

Fig. 5. VirusTotal Hash Scan Result for the Legitimate Stub [27]

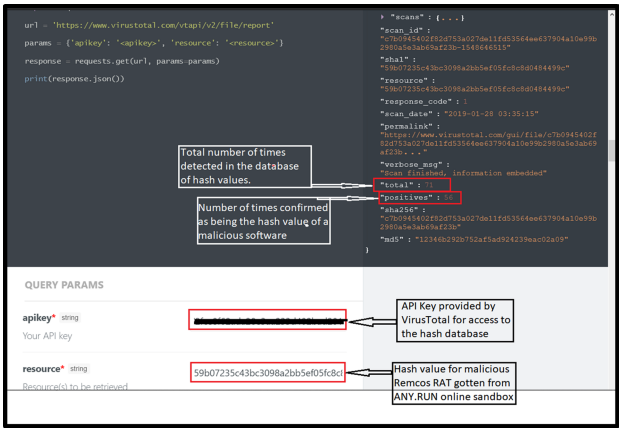

Fig. 6. VirusTotal Hash Scan Result for the Malicious Stub [27]

V. Discussion

A. Results

The results section began with a comparative analysis of some open-source RAT Detection tools. This analysis was beneficial in identifying the various aspects needed for the source code. A high-level representation of the deficiencies of the identified detection tools was provided through the gap analysis in Table IV. The summation of the identified gaps points out the absence of a host-based, process hash-checking functionality. This absence is the reason for the hash-checking section of the source code.

The sections of the source code are as follows:

1) *Listing of all Installed Applications:* Using the *winreg* and *pandas* python modules.

2) *Process Information Gathering:* Using the *psutil*, *datetime*, *time*, *pandas*, and *argparse* python modules.

3) *Process Integrity Checking (Comparison & Hash Scans):* Using the *xlrd*, *requests*, *hashlib*, *csv*, and *xlsxwriter* python modules.

Table V summarizes the detection results from the initial comparison scan, highlighting the need for the hash-calculating portion of the source code. The logic behind the hash-calculating section is that in a case where a RAT bears the same filename as a legitimate remote access stub, the hash values of both files will be calculated and compared to the hash values of the original stub as calculated and prerecorded in the system administrator's "RAT_Detection_Tool (Admin).xlsx" file. The erring hash value will also be checked against the VirusTotal database, and if present in or absent from the database a result will be returned - "0" identifying it as non-malicious/unknown or "1" signifying it as malicious. The hash scan results are summarized in Table VI. It should be noted that due to the restrictions placed on the VirusTotal developer account used by the researcher, an error will be thrown up if any of the Application Programming Interface (API) allowances in Table VII is exceeded, as provided by VirusTotal:

TABLE VII. API Quota

Domain	Restrictions
Database Access Level	Public (Available for 30 days)
Request rate	1,000 requests/minute
Daily quota	20,000 requests/day
Monthly quota	600,000 requests/month

B. Limitations

In the development of this research, the lack of a real-life environment for testing the implementation's effectiveness was a hindrance to the detection tool's further development. The dynamic nature of the ICS threat landscape is an ideal opportunity for the cognitive development of detection filters. The inaccessibility of functioning ICS was a limitation to the research.

Proprietary restrictions on various components across ICS sectors and service providers also create a challenge for cross-compatibility across the different ICS implementations. These proprietary restrictions also provided a limitation to this research.

The lack of free actual samples of malicious RATs hindered the research from being done on live samples. This hindrance led to the use of file hash values when carrying out one of the research simulations. It also made it impossible to analyze malware patterns which are necessary for creating Yara-python descriptions of malware families using textual or binary patterns.

C. Future Research

The lack of free actual samples of malicious RATs can be overcome by directly liaising with malware research companies like VirusTotal, AnyRun, etc. for the controlled provision of malware samples.

The dynamic nature of the ICS threat landscape is an ideal opportunity for the cognitive development of detection filters. For future research, researchers can test RAT detection tools in a real-life environment.

VI. Conclusion

This research presented a host-based tool for the detection and elimination of RATs in ICS. The RAT-detection tool was encoded in the python programming language and was configured to utilize the Yara-python library for malware definition as is also used by VirusTotal [24]. Although no malware family was analyzed and used to create new Yara rules, the tool can still identify RATs and compare them to a database.

This research identified and addressed the unavailability of a host-based, cross-platform, open-source, and process-integrity checking solution that can detect malicious processes in a system.

The solution also followed the high cohesion & low coupling recommendation for modular software development

by utilizing a single script. This leaves little room for attackers to exploit interconnection points in the mechanism.

Based on the results of the research simulations it is evident that filenames are not sufficient in distinguishing stubs for legitimate remote access tools from stubs for RATs. This insufficiency highlights the need for the hash-calculating aspect of this research.

VII. RECOMMENDATIONS

As the script has been written to run once and only when initiated by the system administrator, it is recommended that the script be configured to run three times daily - once at the beginning of daily operations, once during on-peak hours, and once during off-peak hours. However, it can also be run as many times as required by the system administrator. This configuration should be done either from inside the source code or using a batch file.

This research involved the incorporation of fields like control engineering, computer science, and information systems. For the adequate coverage of such encompassing research, a group of students should be tasked with further research, development, and testing along the lines of creating a more in-depth scan of processes.

The lack of free actual samples of malicious RATs can be overcome by directly liaising with malware research companies like VirusTotal, AnyRun, etc. for the controlled provision of malware samples for research purposes.

VIII. REFERENCES

[1] Trend Micro Incorporated, "Industrial Control System," 9 June 2020. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/definition/industrial-control-system.

[2] T. Seals, "ThreatList: Attacks on Industrial Control Systems on the Rise," 7 September 2018. [Online]. Available: https://threatpost.com/threatlist-attacks-on-industrial-control-systems-on-the-rise/137251/#:~:text=The%20threats%20include%20cryptomining%2C%20ransomware,Spectre%2FMeltdown%20class%20of%20vulnerabilities..

[3] SolarWinds Worldwide, "What Is RAT? Best Remote Access Trojan Detect Tools," 7 February 2020. [Online]. Available: https://www.dnsstuff.com/remote-access-trojan-rat.

[4] F. Imam, "Malware spotlight: What is a Remote Access Trojan (RAT)?," 3 December 2019. [Online]. Available: https://resources.infosecinstitute.com/malware-spotlight-what-is-a-remote-access-trojan-rat/#gref.

[5] T. Alladi, V. Chamola and S. Zeadally, "Industrial control systems: Cyberattack trends and countermeasures," Computer Communications, vol. CLV, pp. 1-8, 1 April 2020.

[6] J. Cowan, "Energy sector tops list of US industries under cyber attack, says Homeland Security report," 12 March 2015. [Online]. Available: https://www.iot-now.com/2015/03/12/30962-energy-sector-stays-top-of-the-list-of-us-industries-under-cyber-attack-says-homeland-security-report/.

[7] T. Menze, "The state of industrial cybersecurity," ARC Advisory Group GmbH & Co. KG, Dusseldorf, 2019.

[8] W. Knowles, D. Prince, D. Hutchison, J. F. P. Disso and K. Jones, "A Survey of Cyber Security Management in Industrial Control Systems," International Journal of Critical Infrastructure Protection, 2014.

[9] O. E. Idrissi, A. Mezrioui and A. Belmekki, "Cyber Security challenges and Issues of Industrial Control Systems - Some Security Recommendations," in IEEE International Smart Cities Conference (ISC2), Casablanca, 2019.

[10] B. Babu, T. Ijyas, M. P. and J. Varghese, "Security Issues in SCADA based Industrial Control Systems," in 2nd International Conference on Anti-Cyber Crimes (ICACC), Abha, 2017.

[11] X. Fan, K. Fan, Y. Wang and R. Zhou, "Overview of Cyber-Security of Industrial Control System," in International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC), Shanghai, 2015.

[12] D.-Y. Kim, "Cyber Security Issues Imposed on Nuclear power plants," Annals of Nuclear Energy, pp. 141-143, 2013.

[13] J. Graham, H. Jeffrey and N. John, "Improving Cybersecurity for Industrial Control Systems," in IEEE 25th International Symposium on Industrial Electronics, Santa Clara, 2016.

[14] E. Montalbano, "Spread of Coronavirus-Themed Cyberattacks Persists with New Attacks," 6 March 2020. [Online]. Available: https://threatpost.com/coronavirus-themed-cyberattacks-persists/153493/.

[15] S. Gatlan, "As Coronavirus Spreads, so does COVID-19 Themed Malware," 27 February 2020. [Online]. Available: https://www.bleepingcomputer.com/news/security/as-coronavirus-spreads-so-does-covid-19-themed-malware/.

[16] D. Testa, M. F. Lepore, A. Pirozzi and L. Mella, "New Cyber Attack Campaign Leverages the COVID-19 Infodemic," 2 February 2020. [Online]. Available: https://yoroi.company/research/new-cyber-attack-campaign-leverages-the-covid-19-infodemic/.

[17] D. Jiang and K. Omote, "An Approach to Detect Remote Access Trojan in the Early Stage of Communication," in 29th International Conference on Advanced Information Networking and Applications, Gwangiu, South Korea, 2015.

[18] S. C. Pallaprolu, J. M. Namayanja, V. P. Janeja and C. Adithya, "Label Propagation in Big Data to Detect Remote Access Trojans," IEEE International Conference on Big Data (Big Data), pp. 3539-3547, 2016.

[19] S. Wu, S. Liu, W. Lin, X. Zhao and S. Chen, "Detecting Remote Access Trojans through External Control at Area Network Borders," ACM/IEEE Symposium on Architectures for Networking and Communications, pp. 131-141, 2017.

[20] M. Mimura, Y. Otsubo, H. Tanaka and H. Tanaka, "A Practical Experiment of the HTTP-Based RAT Detection Method in Proxy Server Logs," in 12th Asia Joint Conference on Information Security (AsiaJCIS), Seoul, 2017.

[21] H. Zhu, Z. Wu, J. Tian, Z. Tian, H. Qiao, X. Li and S. Chen, "A Network Behavior Analysis Method to Detect Reverse Remote Access Trojan," in 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2019.

[22] M. Mimura, Y. Otsubo and H. Tanaka, "Evaluation of a Brute Forcing Tool that Extracts the RAT from a Malicious Document File," in 11th Asia Joint Conference on Information Security, Fukuoka, 2016.

[23] A. A. Awad, S. G. Sayed and S. A. Salem, "Collaborative Framework for Early Detection of RAT-Bots Attacks," IEEE Access, vol. 7, pp. 71780-71790, 2019.

[24] VirusTotal, "Yara: The pattern matching swiss knife for malware researchers (and everyone else)," 4 November 2020. [Online]. Available: https://virustotal.github.io/yara/.

[25] M. Gregg and R. Johnson, Certified Information Systems Auditor (CISA) Cert Guide, Indianapolis, Indiana: Pearson Education, Inc., 2018, pp. 204-205.

[26] ANYRUN, "Public Submissions," 13 November 2020. [Online]. Available: https://app.any.run/submissions/#tag:remcos.

[27] VirusTotal, "API Responses," 21 November 2020. [Online]. Available: https://developers.virustotal.com/reference#file-report.