SECURING REMOTE ACCESS NETWORKS USING MALWARE DETECTION TOOLS FOR INDUSTRIAL CONTROL SYSTEMS

Okechukwu Ude

Project report

Submitted to the Faculty of Graduate Studies,
Concordia University of Edmonton

in Partial Fulfillment of the
Requirements for the Final
Research Project for the Degree

MASTER OF INFORMATION SYSTEMS ASSURANCE MANAGEMENT

Concordia University of Edmonton
FACULTY OF GRADUATE STUDIES
Edmonton, Alberta

December 2020

SECURING REMOTE ACCESS NETWORKS USING MALWARE DETECTION TOOLS FOR INDUSTRIAL CONTROL SYSTEMS

Okechukwu Ude

Approved:

Bobby Swar [Original Approval on File]

Bobby Swar Date: December 8, 2020

Primary Supervisor

Edgar Schmidt [Original Approval on File]

Edgar Schmidt, DSocSci Date: December 14, 2020

Dean, Faculty of Graduate Studies

Table of Contents

List of Tables

List of Figures

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. Like every other facet of life, ICS have witnessed the integration of information systems (IS) into their processes, connecting them to the much larger Internet of Things (IoT). Despite several significant advancements--such as controlled-environment agriculture, automated train systems, and smart homes, achieved in critical infrastructure sectors through the integration of IS and remote capabilities with ICS, the fact remains that these advancements have also introduced vulnerabilities that were previously either non-existent or negligible, one being Remote Access Trojans (RATs). RATs are a crucial tool for every advanced persistent threat (APT) attack mounted against any organization, and as such attackers put a lot of effort into making them as undetectable as possible. 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: ICS, information technology, operational technology, remote access trojans, malware detection, Purdue control hierarchy, hash calculation, confidentiality, integrity, availability, RATs

Introduction

Background

The term “Industrial Control System” is used to refer to any combination of devices, systems, networks and controls with the purpose of operating or automating industrial processes (Trend Micro Incorporated, 2020). With such a description, the important and indispensable role of ICS in society can be seen in almost every industry. From healthcare, all the way to finance, ICS are central to daily operations and are built to function differently depending on the industry.

ICS serve as a bridge between the physical world of organizations, key societal services, critical infrastructure and the world of ICT (Luiijf, 2016). Due to the significant role of ICS in society, any threat to the functioning of ICS can be perceived as a threat to society. They can be perceived as:

- threats to the objectives of a business due to ICS-related organizational aspects.
- architectural and technological threats like old technology, insecurity by design etc.
- networking and telecommunication threats as a result of operational environments, dependencies of ICT systems, direct connection to the internet, and remote access capabilities.

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--a trend that is on the

increase. While the main source of these infections is the internet via phishing emails, the threats introduced include crypto mining, ransomware, remote access trojans and a host of others (Seals, 2018).

Remote access trojans are a rampant threat that have become synonymous with major cyberattacks. Initially, RATs referred to Remote Access Technologies, but overtime have become more commonly used to refer to Remote Access Trojans--a more malicious variant of remote access technologies (SolarWinds Worldwide, 2020). Remote access trojans are a form of malware which grant 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 (Imam, 2019).

Presently, adequately patched and regularly updated operating systems, browsers & antivirus software, other physical and logical controls work at protecting the host systems while IDS/IPS work at monitoring the network and/or hosts for RAT activity. As robust as protective measures are, remote access trojans 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 (Alladi et al., 2020). According to a Department of Homeland Security report, at least 55% of the 245 reported ICS attack cases in 2015 were attributed to RATs (Cowan, 2015). 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 (Menze, 2019). As such, increasing the detection strength of

current malware-detection implementations would reduce the likelihood of data breaches, leading to financial and reputational risk optimizations.

The research project will 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 will be encoded in the python programming language and will utilize the yara-python library in line with cybersecurity best practices. This research project detects RATs on a host system but unlike antivirus software--which straightaway assess applications against databases of known threats, this research project analyzes process patterns of running applications and processes, then flags suspicious applications or processes for further analysis.

The Introduction Section--where ICS and ICS vulnerabilities are broached together with a definition of the key problem and a recommended solution, marks the beginning of this research paper. The Literature Review Section discusses some related works, throwing more light on the differences between this method and previous methods. In the Methodology Section, the scope, questions and solution procedures raised by the project are to be described. A summary of the project's simulation within a controlled environment will be presented in the Results Section. Finally, the method's results, limitations and opportunities will be reviewed in the Discussion Section.

Literature Review

Industrial Control Systems

According to Cai (2008, as cited in Foreman et al., 2012), over the ages, ICS have been designed and implemented as isolated, patented and air-gapped systems with no

attachment to the outside world. Recent technological advancements--like computers, network and process integration, by permeating the various critical infrastructure sectors have brought about some far-reaching benefits.

Notwithstanding, the incorporation of cyber-capabilities together with legacy systems have also introduced some formerly non-existent or negligible vulnerabilities into industrial control systems, leaving ICS open to all sorts of cyber-attacks. Attacks like Iran's Stuxnet of 2010, Bahrain's Dustman of 2019 and Saudi Arabia's Triton of 2017 are just a few out of 400+ reported cyber-attacks on ICS since 2006 (Center for Strategic & International Studies, 2020).

ICS categorization. In addressing ICS cyber-vulnerabilities, a better understanding of ICS categories would facilitate an adequate analysis of ICS-generic threat surfaces.

By sector. In the *Homeland Security Presidential Directive 7* (HSPD-7) policy, the George W. Bush-led administration categorized ICS into 17 critical infrastructure sectors (Office of the Press Secretary, 2003). In the *Presidential Policy Directive-21* (PPD-21) document 10 years later, the Obama-led administration categorized ICS into 16 critical infrastructure sectors, replacing "National Monuments & Icons" and "Postal & Shipping" with "Critical Manufacturing" (The White House Office, 2013). These classifications are shown in Table 1:

Table 1

Critical Infrastructure Sectors that Utilize ICS

S/No.	HSPD-7[a]	PPD-21[b]
1	Agriculture and food	Food and agriculture
2	Defense industrial base	Defense industrial base
3	Energy	Energy
4	Public health and health care	Healthcare and public health
5	Banking and finance	Financial services
6	Drinking water and water treatment systems	Water and wastewater systems
7	Chemical	Chemical
8	Commercial facilities	Commercial facilities
9	Dams	Dams
10	Emergency services	Emergency services
11	Nuclear reactors, materials, and waste	Nuclear reactors, materials, and waste
12	Information technology	Information technology
13	Communications	Communications
14	Transportation systems	Transportation systems
15	Government facilities	Government facilities
16	National monuments and icons	Critical manufacturing
17	Postal and shipping	

[a.] Office of the Press Secretary (2003). [b] The White House Office (2013).

By function. Ackerman (2017) explains that ICS are a variety of control systems/sub-systems utilized industrially in production technology. These sub-systems work together in a collaborative manner to achieve a common goal and can be categorized into three groups according to their functions as summarized in Table 2:

Table 2

ICS Categorization Based on Function

S/No.	Category	Description
1	View Function	This functionality refers to the ability to see the current state of an ICS in real time.
2	Monitor Function	This functionality keeps an eye on critical values - like temperature and pressure, contrasts their present values with preset threshold values, and issue alerts based on the results:

S/No.	Category	Description
3	Control Function	This functionality refers to the processes behind controlling, moving, activating and initiating of actuators, valves and motors

By use. According to Sullivan et al. (2016), ICS can be categorized based on their usage and based on the physical/geographical location of the supervisory components--like Human Machine Interface (HMI) and Data Historian, relative to the controller. Sullivan et al. (2016) group them as shown in Table 3:

Table 3

ICS Categorization Based on Usage

S/No.	Category	Description
1	Safety Instrumented Systems	Responsible for monitoring automation processes and actively preventing an unsafe plant state or operations.
2	Distributed Control Systems	Responsible for controlling numerous automated processes within a single plant.
3	Building Automation System	Responsible for monitoring and controlling a building's support services like cooling, ventilation, air conditioning, heating etc.
4	Supervisory Control and Data Acquisition	Responsible for data collection and monitoring of automation across vast geographic areas.
5	Process Control Systems	Responsible for controlling an automation process in a manufacturing environment.
6	Energy Management Systems	Responsible for monitoring and controlling electricity generation, transmission and distribution.
7	Other Types of ICS	Referring to compact forms of ICS integrated into less obvious systems like cars, trains, trucks and autonomous systems like robots

Functional components of ICS. Sullivan et al. (2016), note that functional components common to all ICS can be classified into four groups--controllers, communication devices, field devices, and software applications. They are detailed as follows:

Controllers. According to Sullivan et al. (2016), the three important controllers in ICS are as shown in Table 4:

Table 4

ICS Controllers

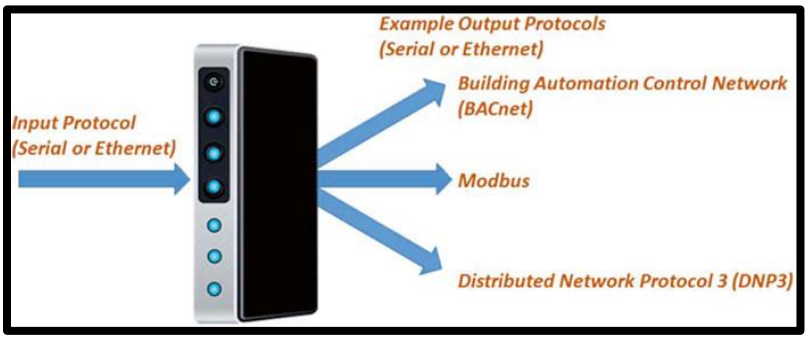
CONTROLLERS		
S/No.	**Name**	**Description**
1	Programmable Logic Controller (PLC)	A microprocessor-controlled device which repeatedly reads sensor measurements, uses them to calculate necessary changes, and effecting the changes using physical machinery. [a]
2	Remote Terminal Unit (RTU)	A device--microprocessor-controlled in nature, which acquires and processes real-time data from a plant and relays the information to the HMI unit in the control room. [b]
3	Intelligent Electronic Device (IED)	A device that executes electrical protection functions, local control intelligence, process monitoring, and direct communication with a SCADA system.

[a] (McLaughlin & Zonouz, 2014). [b] (Shang, 2010).

Communication Devices. Sullivan et al. (2016) classify *Engineering Workstations*--a server or desktop computer containing a standardized operating system, *a Front-end Processor (FEP)*, and *Communication Gateways* as ICS communication devices. An example of a communication gateway is shown in Figure 1:

Figure 1

Communication Gateway (Sullivan et al.,2016)



***Field Devices*.** Consisting of actuators, transducers, sensors, and other machinery which connect directly to a controller through a communication gateway (Sullivan et al., 2016).

***Software Applications*.** Sullivan et al. (2016), mention the *Human Machine Interface* and *Data/Operational Historian* as software applications utilized in ICS.

ICS Security Architecture

With the vast nature of the ICS threat landscape, there is no surefire method of implementing security that will guarantee 100% confidentiality, integrity and availability of Operational Technology (OT) and Information Technology (IT) assets. In reviewing ICS cybersecurity implementations, the following model, frameworks and standards are to be discussed:

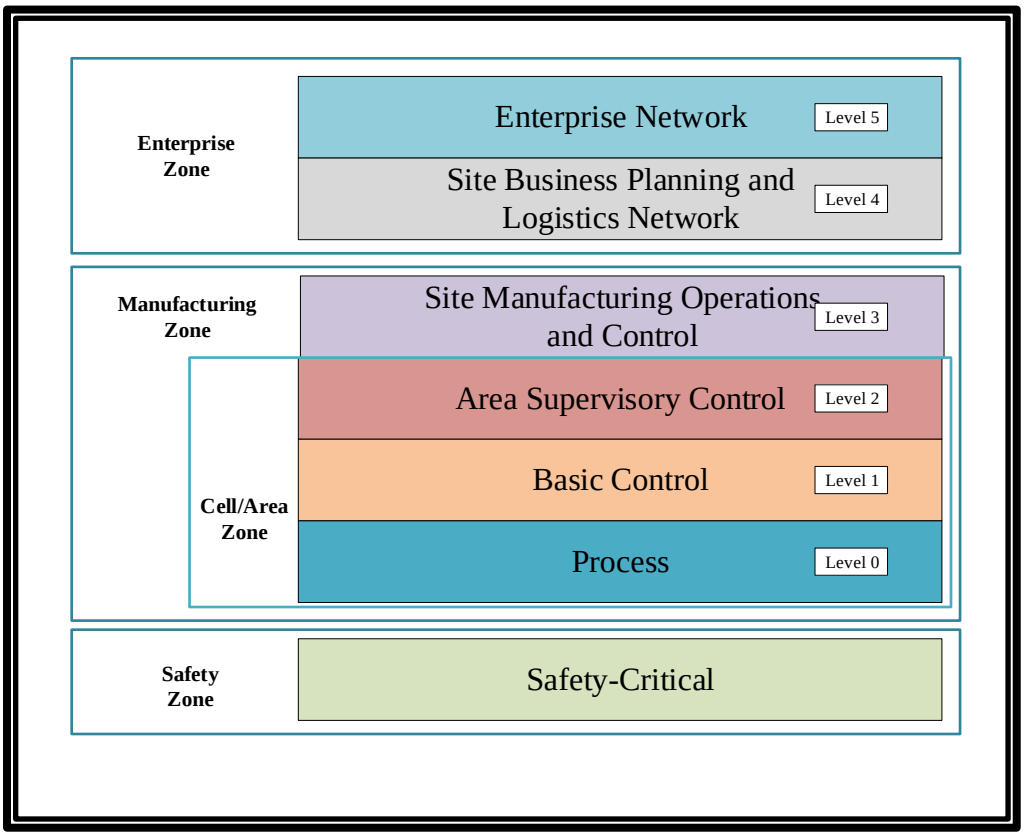
Purdue model for control hierarchy. Adopted from the Purdue Enterprise Reference Architecture (PERA) model which was created from the International Society of Automation (ISA) ISA-99 set of standards, the Purdue model is an industry-accepted model which visualizes the interdependencies and interconnections between ICS components, while encouraging network segmentation as a security measure for ICS (Ackerman, 2017). Rockwell Automation (2008) describes the Purdue Model as a familiar and well-understood model used in the manufacturing sector, which groups devices as well as equipment in a top-down manner according to their functions.

According to Obregon (2015), the Purdue model uses the idea of zones to logically partition an enterprise and ICS networks' components into subdivisions according to similarities in functions or requirements. Rockwell Automation (2008)

maintains a similar view but portrays the “Cell/Area Zone” as a subsection of the “Manufacturing Zone” as shown in Figure 2:

Figure 2

Six-Level Plant Architecture



Usually separated from the rest of the control system by air gaps and excluded from the six levels, the safety zone is seen as “the highest priority function in industrial automation and control systems” (Rockwell Automation, 2008).

Ackerman (2017), Obregon (2015) and Rockwell Automation (2008) add that the Purdue model also subdivides the ICS architecture into three zones--enterprise, industrial demilitarized and manufacturing zones as detailed in Table 5:

Table 5

ICS Zones and Levels Descriptions

Level	Name	Description
ENTERPRISE ZONE		
5	Enterprise Network	Domain for centralized information technology (IT) systems and functions i.e. business-to-customer and business-to-business service systems, enterprise resource planning. Systems and functions at the enterprise level are managed directly by the IT organization. [a]
4	Site Business Planning and Logistics Network	Contains IT systems that handle maintenance and operational management, communication (e-mail and phone), printing services, scheduling, reporting, inventory management and capacity planning. [b]
INDUSTRIAL DEMILITARIZED ZONE (IDMZ)		
Ackerman (2017) attributes the IDMZ to the efforts put into the creation of security standards like the National Institute of Standards and Technology Cybersecurity Framework (NIST CSF) and the North American Electric Reliability Corporation critical infrastructure protection plan (NERC CIP). The IDMZ is a layer resulting from the sharing of information between the IT systems of the enterprise zone and the operating technology (OT) systems of the manufacturing zone. [c]		
MANUFACTURING/INDUSTRIAL ZONE		
3	Site Manufacturing Operations and Control	Systems here are responsible for control plant operations management aimed at producing the desired product. Comprised of engineering workstations, network file servers, remote access services, DNS, DHCP, Active Directory, NTP etc. [b]
2	Area Supervisory Control	Containing functions and systems like level 3 but associated with runtime supervision and production facility operations for specific areas. [a]
1	Basic Control	Domain of all controlling equipment. Responsible for opening valves, moving actuators, starting motors. Comprised of PLCs, variable frequency drivers (VFDs), dedicated proportional-integral-derivative (PID) controllers etc. [c]
0	Process	Performs the basic functions of ICS e.g. driving a motor, measuring variables, output setting, welding etc. Comprised of sensors, actuators and other manufacturing-involved devices. [a]

[a] (Rockwell Automation, 2008). [b] (Obregon, 2015). [c] (Ackerman, 2017).

Recommended frameworks, standards and guidelines. Securing ICS networks is one task which requires a substantial amount of work. Not only from the administration of the network alone but from the state authorities as well to enforce relevant security policies and procedures. When policies are implemented, attention is required for possible changes that will be needed (Panayotidis, 2019).

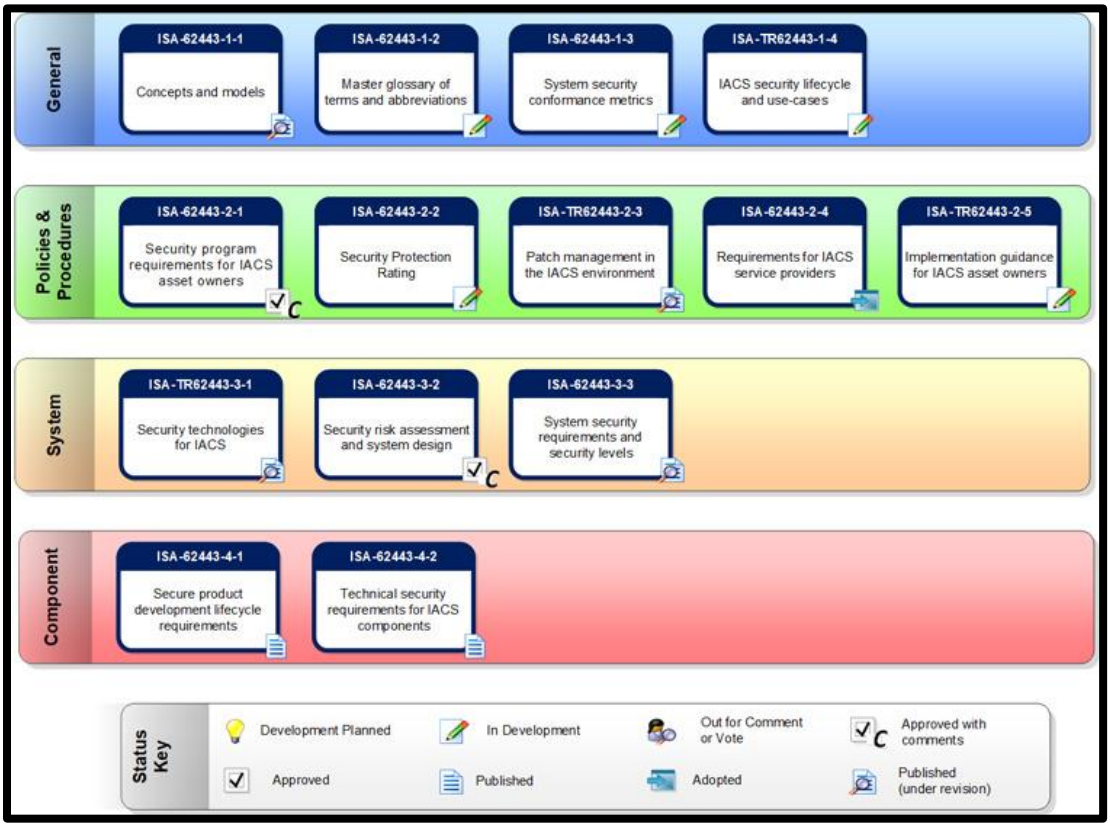
This section will highlight some relevant frameworks, some relevant standards and recommended practices/guidelines which are advantageous in securing ICS.

Recommended Frameworks. An International Society of Automation (ISA) ISA99-committee development imitated by the International Electrotechnical Commission (IEC), the *ISA/IEC 62443* standards series is a pliable framework designed to label and alleviate present and future security vulnerabilities in ICS (The International Society of Automation, 2018).

Figure 3 illustrates the various components of the ISA/IEC 62443 series:

Figure 3

ISA/IEC 62443 series (The International Society of Automation, 2018)



Note. The phrase "Industrial Automation and Control Systems (IACS)" can be used interchangeably with ICS.

As further evidence of ISA/IEC 62443's widespread acceptance, the United Nations Economic Commission for Europe (UNECE) affirmed plans to merge the ISA/IEC 62433 series with the UNECE's Common Regulatory Framework on Cybersecurity (CRF) (The International Society of Automation, 2019).

In the ISA/IEC 62443 series, some key standards are worthy of note as seen in Table 6 (Arampatzis, 2019):

Table 6

ISA/IEC 62443 Key Standards

S/No.	Standard	Description [a]
1	IEC 62443-2-4	Specifies a requirement set for security capabilities of service providers in ICS industries. During the integration and maintenance of a control solution, the 62443-2-4 security capabilities can be offered to the ICS management.
2	IEC 62443-4-1	Specifies process requirements thereby facilitating the secure development of products for ICS usage.
3	IEC 62443-4-2	Provides technical details for ICS component requirements (CRs) relating to the seven foundational requirements as described in IEC 62443-1-1.
4	IEC 62443-3-3	Control system requirements (SRs) are detailed in a technical manner relating to the seven foundational requirements as described in IEC 62443-1-1.

[a] (International Society of Automation, 2018).

Another well-used and highly recommended framework for ICS is the NIST Framework for Improving Critical Infrastructure Cybersecurity (also known as NIST CSF). Focusing on guiding cybersecurity activities by using business guides, the NIST CSF considers cybersecurity risks as a constituent of organizational risk management processes (Technology, 2018).

Recommended Standards. The National Institute of Standards and Technology (NIST) is a non-regulatory federal agency within the U.S. Department of Commerce formed with the aim of promoting U.S. industrial competitiveness through innovative advancements in the science of measurement, standards and technology, in a manner that amplifies quality of life and economic security (Wicked Problem Perspectives Working Group, 2020). Several standards created by NIST have gone into the world and become guidelines in almost every critical infrastructure sector. In ICS, the *NIST Special*

Publication 800-82 is the standard for ICS security. Titled *"Guide to Industrial Control Systems (ICS) Security"*, the publication serves as a guide towards securing ICS. It addresses the performance, reliability and safety requirements of ICS (Stouffer et al., 2015). It also provides an overview of ICS and its system topologies, identifying organizational mission threats and ICS-dependent business functions, highlights known ICS vulnerabilities, and recommends adequate security controls to respond to related risks (Quanterion Solutions Incorporated, 2015).

Knowles et al. (2014) mention a series of standards which although general in scope, provide for the addressing of security as relates to industrial control systems. Another series of standards in information security, the *ISO/IEC 27000* serves as guidance towards the implementation of an effective information security management system. Among the ISO/IEC 27000 standard family, one child standard that has seen widespread usage by ICS operators is the ISO/IEC 27002, responsible for outlining security controls grouped based on their control objectives (Knowles et al., 2014). Knowles et al. (2014) also report that among the ISO/IEC 27000 family the most notable standard is the ISO/IEC 27019:2013--based on ISO/IEC 27002:2013, which provides supporting information security management system guidelines.

Recommended Guidelines. Knowles et al. (2014) highlight a U.S. Department of Homeland Security (DoHS) document titled "Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies". The DoHS publication discusses technical and managerial strategies for ICS, establishing a security-centric culture using operational security, remote access configuration and management, and recommended patch management practices (Knowles et al., 2014).

Security Issues in ICS

Knowles et al. (2014) explain that giving security of ICS a low priority has been the default perspective 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. (2014) 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:

CIA vs. AIC. The dimensions of the CIA triad are conceptually equally important. According to Knowles et al. (2014) some aspects of the CIA triad are deemed more expedient than others for reasons which could be economical or strategic in nature. 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 (Knowles et al., 2014).

Convergence of IT & OT. Idrissi et al. (2019) 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. (2019) map the rate of increase in vulnerabilities in Table 7:

Table 7

Number of Vulnerabilities between 2015 and 2018

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. The compounding factor is the dearth of security professionals within organizations--both in-house staff and third-party vendors. The fault does not lie with cybersecurity skills gap in the IT sector alone but also with the lack of experience in operational technology environments by available information security professionals (Maddison, 2018).

Communication Protocols in IT & OT. Idrissi et al. (2019), Babu et al. (2017), and Fan et al. (2015) report that the traditional isolated nature of ICS is the reason why ICS communication protocols do not take access control policies such as 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. (2017) add that the failure to build access control via authentication into the ICS design does not mean that authentication and encryption cannot be used with ICS systems.

Malware. Fan et al. (2015) 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 malwares 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 (Kim, 2013). Kim (2013) 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.

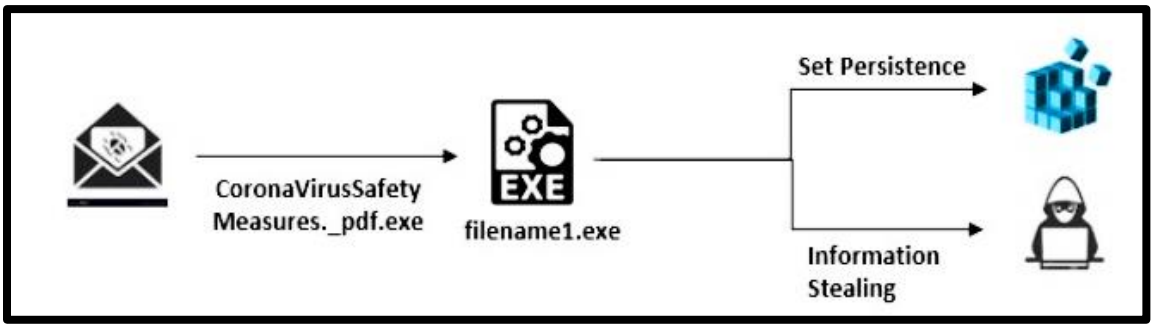
Remote access trojans. In securing industrial control systems, remote access to the control systems is a necessary functionality which facilitates quick troubleshooting and easy configuration irrespective of geographic location (Graham et al., 2016). Remote Access Trojans are a variant of trojans--a type of malware, which exploit the need for remote access functionalities by ICS (SolarWinds Worldwide, 2020). Remote access trojans are a form of malware which grant 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 (Imam, 2019).

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 (Montalbano, 2020). Gatlan (2020) 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\RunOnce"*. 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. Figure 4 shows an illustration of the Remcos attack process:

Figure 4

Remcos RAT Attack Chain (Testa et al., 2020)



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). The power of Remote Access Trojans is more instrumental when applied in an APT type of attack. RATs are ideal for APTs because not only do RATs not slow down a computer's performance or automatically delete files upon installation, but because RATs are quite adaptable--just like live rats (SolarWinds Worldwide, 2020).

Efforts have been made at creating tools & methods for the detection of these RATs. Wu et al. (2017) identify five categories of RAT-Detection methods as shown in Table 8:

Table 8

Classification of RAT-Detection Methods

Categories	Subclass	Applied to RATs Detection
Host-based	Signature	Static, detect binary
	Behavior	Sandbox, dynamic detect
Network-based	Signature	Static, detect application layer
	Anomaly	Statistics of traffic flow
	Protocol	Non-standard protocol

The methods in Table 9 fall under the network-based detection category while the methods in Table 10 fall under the host-based detection category:

Table 9

Research Done on Network-Based RAT Detection

Researchers	Network-Based Methods
Jiang & Omote (2015)	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)	An ensemble-based label propagation technique that detects RATs in huge, processed, and unlabeled datasets using a big data engine.
Wu et al. (2017)	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)	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 behaviours.
Zhu et al. (2019)	Six unsupervised classification algorithms are implemented on a network traffic dataset on the basis of the states of four uncorrelated network activity features

Researchers	Network-Based Methods
	obtained from TCP sessions to identify and illustrate the difference between RAT-traffic and legitimate sessions.

Table 10

Research Done on Host-Based RAT Detection

Researchers	Host-Based Methods
Mimura et al. (2016)	A brute forcing tool is developed and used to overcome obfuscation while disinfecting a RAT-infested document file.
Awad et al. (2019)	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. Among the host-detection options, the closest to a precisely host-based RAT-detection prototype was the host agent by Awal et al. (2019). Awal et al. (2019) created an anomaly detection policy which would be enforced once any suspicious activity sequence was initiated.

Methodology

This research project presents a host-based tool for the detection and elimination of RATs in ICS. The RAT detection tool is based on 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 (VirusTotal, 2020).

This research project identifies and addresses some of the weaknesses in the detection capacities of present open source malware-detection tools used for ICS.

Placing an emphasis on the detection of Remote Access Trojans, a Cpython implementation of Python has been developed based on the C/C++ based ClamAV antivirus engine source-code.

This implementation follows the *high cohesion & low coupling* recommendation for modular software development (Gregg & Johnson, 2018).

The research project addresses the following questions:

- What effective and efficient methods were formerly developed to address the problem of Remote Access Trojans 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. These RAT detection tools were identified based on the capability of working on a host system and a detection rate of at least 80%. The effectiveness and efficiency of identified methods were deduced based on the tool's detection rates of test RAT samples used in the research project.

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. Some samples of available RAT-detection methods were identified and evaluated based on:

- their ability to work on a host system
- their possession of a detection rate of at least 80% of the RAT samples used in this research project.
- their open-source status.

Equation 1 is a simplified representation of the algorithm for the source code based on the identified gaps:

Equation 1

Code Algorithm

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 VirusTotal database.
        i) If there is a match, kill the app(s) and/or process(es), log the detection and alert the system administrator.
        ii) 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 next scan.

Results

Evaluation & Analysis

A GAP analysis approach was used to identify the deficiencies of some open source tools, and control algorithms were developed into source code based on this analysis. Table 11 below summarizes the attributes of the RAT-detection tools:

Table 11

Some Open-source RAT Detection tools

Tool	Features	
AIDE (Advanced Intrusion Detection Environment)	• supported hash functions: md5, sha1, rmd160, tiger, crc32, sha256, sha512, whirlpool (additionally with libmhash: gost, haval, crc32b) [a] • supports Extended file system attributes, XAttrs, SELinux, and Posix ACL (plain text configuration file compilation is required for this) [a] • supports the Inode, Uid & Gid file types; and the following file attributes: Mtime, Ctime and Atime, Number of links, Block count, Size, Link name [a] • expression support for selective inclusion or exclusion of files and directories that are to be monitored [a] • provided that zlib support is compiled in, gzip database compression is possible [a]	
	Pros	Cons
	• Facilitates file integrity verification by creating database directories as specified in the configuration text file [b] • Hash creation for each file and supports CRC-32, MD5, HAVAL, Tiger, WHIRLPOOL RMD-160, SHA-512, SHA-256, and SHA-1 message digest algorithms [b] • Daily automatic report generation and the report is stored in /var/mail/root [b]	• AIDE reports file alterations after the incident and makes no attempt to prevent the alteration [b] • By default, database directories are left unencrypted, thereby creating a need for additional security measures, else the configuration file could be modified by an attacker [b] • No IPS functionality [b] • Supports only Unix-based operating systems [a]
Samhain	• PCI DSS Compliant [d] • Centralized management [d]	

Tool	Features	
	• File integrity checks [d] • Host integrity monitoring [d] • Log facilities [d] • Integration with other systems/Active response [d]	
	Pros	Cons
	• Supports server logging via encrypted and authenticated connections, database and configuration file signing [b] • Supports incremental checks on growing logfiles [b]	• Modularization makes configuration too complex • No IPS functionality [b]
Snort	Operates in three modes: • Network Intrusion Detection System (NIDS) mode - detects and analyzes network traffic. [e] • Packet Logger mode - logs the packets to disk [e] • Sniffer mode - reads the packets on the network and displays them as a continuous stream on a screen [e]	
	Pros	Cons
	• Ease of deployment. [b] • Well-documented and tested signature set. [b] • Supports numerous operating systems (HP-UX, Tru64, IRIX, BSD, Solaris, MacOS X, Linux, Windows etc.). [b] • IPS functionality can be enabled through the configuration of "inline mode". [b]	• Large rules database requiring enormous processing power to match the packet rate [b] • Packet monitoring within large internetworks is a resource-demanding task [b] • Fragmented packets go undetected within high speed networks (>5Gbps) [b]
OSSEC Host Intrusion Detection System	• Log-based Intrusion Detection (LIDs) [f] • Rootkit and Malware Detection [f] • Active Response [f] • Compliance Auditing [f] • File Integrity Monitoring (FIM) [f] • System Inventory [f]	
	Pros	Cons
	• Can analyze logs from multiple devices and in multiple formats. [b] • Designed as an active response system, capable of monitoring and responding to threats. [b] • IPS functionality [b]	• Version upgrades are problematic due to the overwriting of configurations with every upgrade [b] • Transfer of pre-shared keys before establishment of client-server communication is problematic due to the blowfish algorithm used for encryption. [b]

<table>
<tr><th>Tool</th><th colspan="2">Features</th></tr>
<tr><td rowspan="4">Sagan</td><td colspan="2">Supports the use of the entire CPU cores for the processing of logs in real-time. [g]
Minimal CPU and memory requirements. [g]
Similarity in rule syntaxes makes rule management easy and this is due to its correlation with Suricata/Snort IDS / IPS systems. [g]
Ability to save alerts in Snort’s unified2 binary data format as well as Suricata's json format facilitating a smoother log-to-packet correlation. [g]
Compatibility with GUI security platforms like EveBox, BASE, Snorby, and Sguil. [g]
Simplified data-export process via syslog with other SIEMs [g]
Location-based tracking of events using IP address source or destination data is supported. [g]
Time-of-day based usage monitoring (e.g., creating a rule that is triggered when an administrator logs in at a specific time of day). [g]
Supports various parsing and data extraction methods via liblognorm or built-in options for rule-parsing like parse_dst_ip, parse_src_ip, parse_string, parse_port, parse_hash (SHA1, SHA256, MD5). [g]</td></tr>
<tr><td>Pros</td><td>Cons</td></tr>
<tr><td>Open source [c]
Compatible with Snort data [c]
Multiple third-party integrations [c]
Can be integrated with firewalls for IP blocking [c]</td><td>Works on only Unix-based operating systems
Steep learning curve (many features) [c]</td></tr>
<tr><td rowspan="3">Security Onion</td><td colspan="2">Linux distribution, free and open source, for threat hunting, corporate security monitoring, and log management.
Consisting of Elasticsearch, Logstash, Kibana, Snort, Suricata, Zeek, Wazuh, Sguil, Squert, NetworkMiner, CyberChef, and several other security tools. [h]</td></tr>
<tr><td>Pros</td><td>Cons</td></tr>
<tr><td>Although modularized, the single operating system eases the complexity of configuration</td><td>Acts as a standalone Linux distribution (operating system), which defeats the purpose of RAT host-detection</td></tr>
</table>

[a] (Haugwitz, 2019). [b] (Ambati & Vidyarthi, 2013). [c] (Samson, 2020). [d] (Samhain Labs, 2020). [e] (Cisco, 2020). [f] (OSSEC Project Team, 2020). [g] (Quadrant Information Security, 2020). [h] (Security Onion Solutions, 2020)

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.

Based on the identified gaps, control algorithms were developed into source code and were used to create the research project's source code. The gap analysis is illustrated in Table 12 below:

Table 12

A Gap Analysis of Selected Open-source RAT-Detection Tools

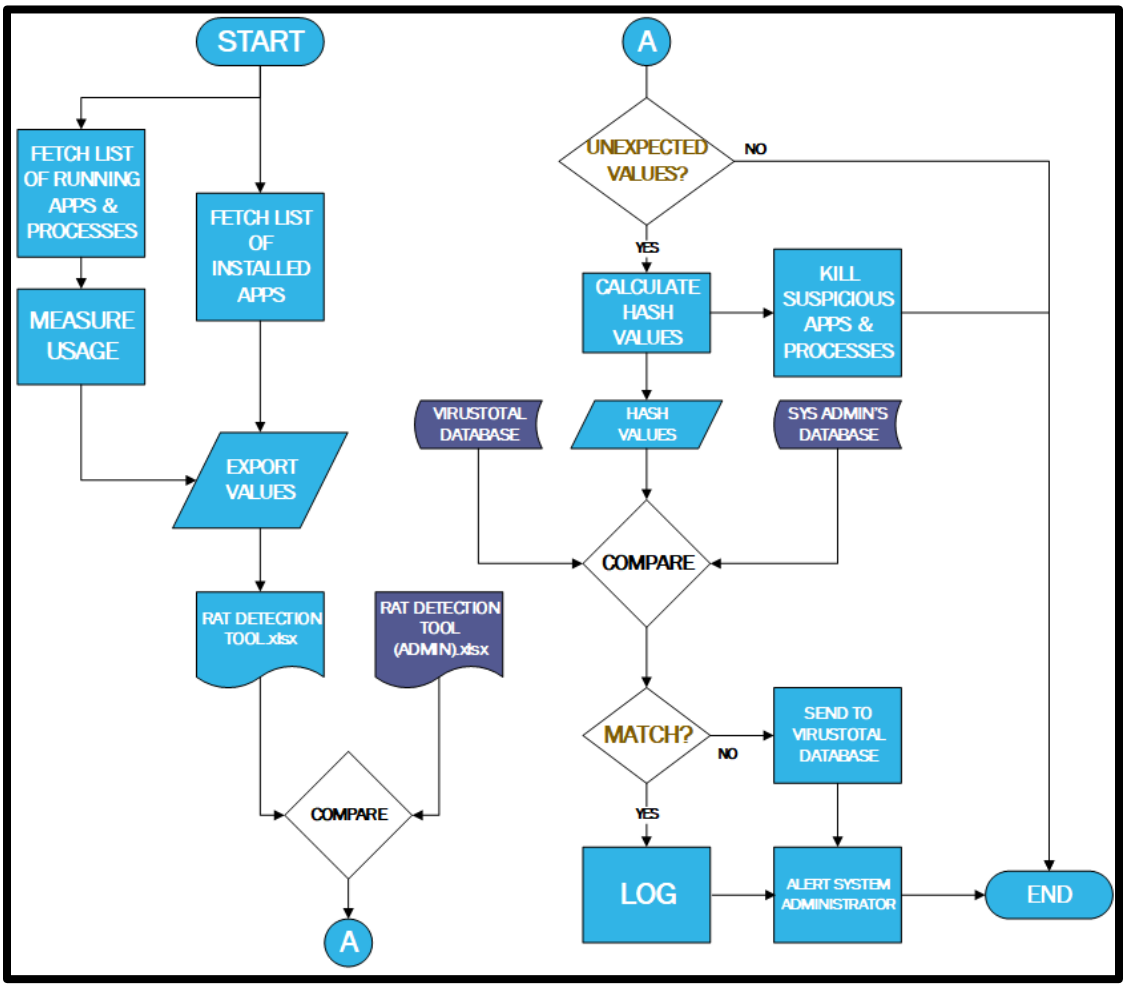
S/No.	RAT-Detection Tool	Present State	Desired State	Gap
1	AIDE (Advanced Intrusion Detection Environment)	• Supports only Unix-based operating systems	Host-based process-integrity checker with cross-platform compatibility	Non-emulated cross-compatibility
2	Samhain	• Host-based process-integrity checker • Works on windows only through "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

S/No.	RAT-Detection Tool	Present State	Desired State	Gap
6	Security Onion	• Standalone Linux distribution	Host-based process-integrity checker with cross-platform compatibility	Non-emulated cross-compatibility

In order to address the gaps in the identified tools, an algorithm was developed and is depicted in Figure 5:

Figure 5

RAT Detection Tool Flowchart



Below are some sections of the source code for the algorithm:

Figure 6

Python Modules and Containers Used

```
import pandas as pd
import psutil
import wmi
import winreg
import xlrd
import time
import argparse
import xlsxwriter

# from os import system, name
from datetime import datetime

start = time.time()  # begin the runtime counter
c = wmi.WMI()
IA_counter = 0  # initialising the counter for the "Installed Applications" code block
APM_counter = 0  # initialising the counter for the "Application & Process Monitor" code bloc
program_timer = 0  # initialising the counter for initiating the "Process Integrity-Checking"
n = 0  # used to determine the number of times to print the display
```

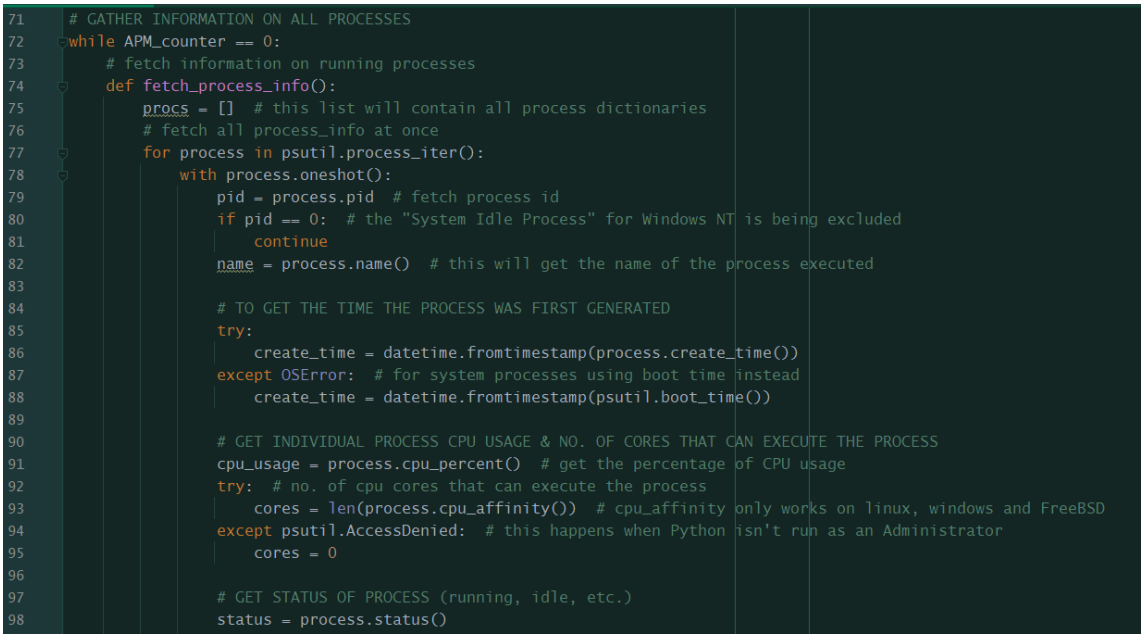
Figure 7

Fetching the List of Installed Applications (Step 1)

```
# 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'
                try:
                    application['publisher'] = winreg.QueryValueEx(asubkey, "Publisher")[0]
                except EnvironmentError:
                    application['publisher'] = 'undefined'
                application_list.append(application)
            except EnvironmentError:
                continue
        return application_list
```

Figure 8

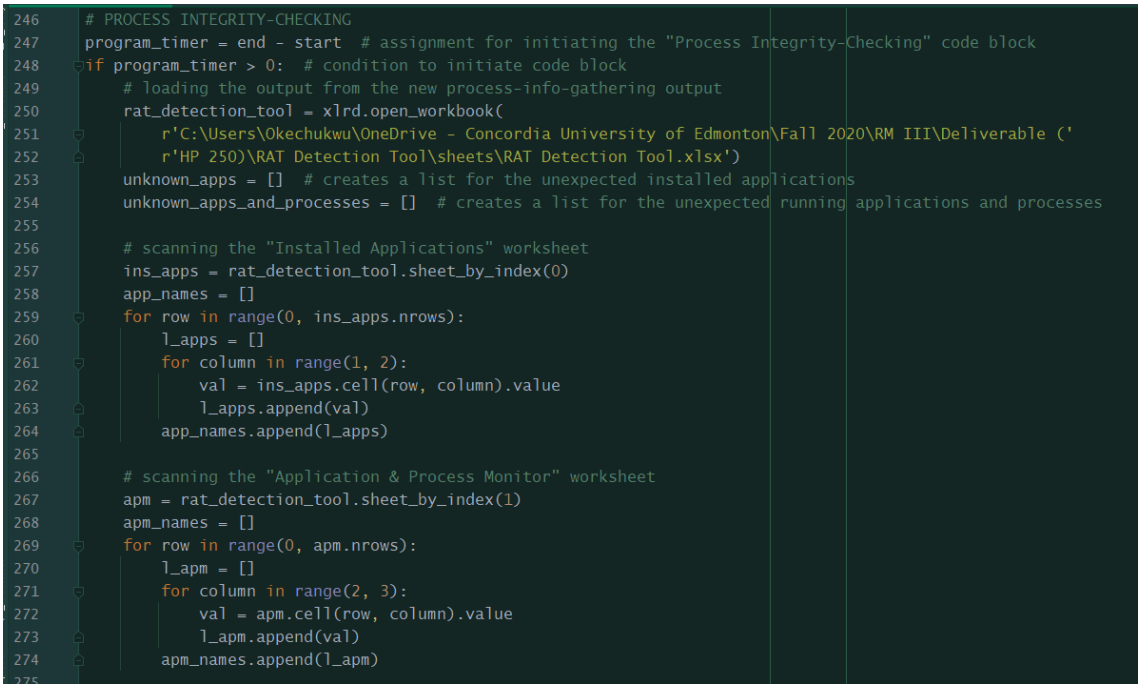
Fetching Information on Running Applications and Processes (Step 2)



```
# 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 excluded
                    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())

                # GET INDIVIDUAL PROCESS CPU USAGE & NO. OF CORES THAT CAN EXECUTE THE PROCESS
                cpu_usage = process.cpu_percent()  # get the percentage of CPU usage
                try:  # no. of cpu cores that can execute the process
                    cores = len(process.cpu_affinity())  # cpu_affinity only works on linux, windows and FreeBSD
                except psutil.AccessDenied:  # this happens when Python isn't run as an Administrator
                    cores = 0

                # GET STATUS OF PROCESS (running, idle, etc.)
                status = process.status()
```

Figure 9

Process-Integrity Checking Function (Step 5)



```
# PROCESS INTEGRITY-CHECKING
program_timer = end - start  # assignment for initiating the "Process Integrity-Checking" code block
if program_timer > 0:  # condition to initiate code block
    # loading the output from the new process-info-gathering output
    rat_detection_tool = xlrd.open_workbook(
        r'C:\Users\Okechukwu\OneDrive - Concordia University of Edmonton\Fall 2020\RM III\Deliverable ('
        r'HP 250)\RAT Detection Tool\sheets\RAT Detection Tool.xlsx')
    unknown_apps = []  # creates a list for the unexpected installed applications
    unknown_apps_and_processes = []  # creates a list for the unexpected running applications and processes

    # scanning the "Installed Applications" worksheet
    ins_apps = rat_detection_tool.sheet_by_index(0)
    app_names = []
    for row in range(0, ins_apps.nrows):
        l_apps = []
        for column in range(1, 2):
            val = ins_apps.cell(row, column).value
            l_apps.append(val)
        app_names.append(l_apps)

    # scanning the "Application & Process Monitor" worksheet
    apm = rat_detection_tool.sheet_by_index(1)
    apm_names = []
    for row in range(0, apm.nrows):
        l_apm = []
        for column in range(2, 3):
            val = apm.cell(row, column).value
            l_apm.append(val)
        apm_names.append(l_apm)
```

Simulation & Testing

To experiment on the effectiveness of the new tool, a test environment was setup 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:

- A folder called “RM”, with a “RAT Detection Tool” folder inside.
- A folder called “sheets” inside the “RAT Detection Tool” folder. This folder contains the following:
  - 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.
  - List of Undesirables.xlsx
  - RAT Detection Tool.xlsx
  - Unknown Processes.csv

Default applications & processes were left installed and running after the initialization of each test virtual machine. For the purposes of 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 below:

Table 13

Results of the Comparison Scan on Tested Sample's Stubs

S/No.	Family Name	Programming 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)
6	CloudNet	.NET	2014	Detected (cloud.exe)
7	Comet RAT	.NET	2015	Detected (server1.exe)
8	Coringa	.NET	2015	Detected (Servidor.exe)
9	Crimson	JAVA	2012	Detected as javaw.exe (Requires hash value to differentiate)
10	ctOs	.NET	2015	Detected (server.exe)
11	CyberGate	Delphi	2010	Detected (server.exe)
12	DarkComet	Delphi	2008	Detected (server.exe)
13	DH RAT	Delphi	2013	Detected (server_ready.exe)
14	D-RAT	.NET	2012	Detected (D-RAT.vhost.exe)
15	Frutas	JAVA	2012	Detected as javaw.exe (Requires hash value to differentiate)
16	Greame	Delphi	2012	Detected (server.exe)
17	HAKOPS	Visual Basic	2014	Detected (server.exe)
18	Imminent Monitor	.NET	2013	Detected (IMserver.exe)
19	Imperium	.NET	2011	Detected (server.exe)
20	jSpy	JAVA	2012	Detected as javaw.exe (Requires hash value to differentiate)
21	KilerRAT	.NET	2015	Detected (GetMoney.exe)
22	L6RAT	.NET	2014	Detected (server.exe)
23	Maus	JAVA	2016	Detected as javaw.exe (Requires hash value to differentiate)
24	Mega	.NET	2014	Detected (Server.exe)
25	MLRAT	.NET	2015	Detected (MLRat.exe)
26	MQ5	.NET	2015	Detected (MQ.exe)
27	NanoCore	.NET	2013	Detected (client.exe)
28	NingaliNet	.NET	2016	Detected (Server.exe)

S/No.	Family Name	Programming Language	Debut Year	Comparison Scan Result
29	NjRAT	.NET	2012	Detected (server.exe)
30	Njworm	Visual Basic	2013	Detected (njworm.exe)
31	NovaLite	Delphi	2011	Detected (Server.exe)
32	Nuclear	Delphi	2003	Detected (Server.exe)
33	Orion	Delphi	2014	Detected (orionserver.exe)
34	Pandora	Delphi	2013	Detected (server.exe)
35	Proton	.NET	2014	Detected (server.exe)
36	Pupy	Python	2015	No process to detect. Runs using reflective dll
37	Quasar	.NET	2014	Detected (server.exe)
38	Rabbit-Hole	Delphi	2015	Detected (server.exe)
39	Revenge	.NET	2016	Detected (Client.exe)
40	Spycronic	Delphi	2010	Detected (server.exe)
41	SpyGate	.NET	2013	Detected (Server.exe)
42	Spy-Net	Delphi	2010	Detected (server.exe)
43	Sub-7	Delphi	1999	Detected (ForHost.exe)
44	Turkojan	Delphi	2003	Detected (server.exe)
45	ucuL	C++	2013	Detected (Server.exe)
46	VanTom	.NET	2014	Detected (Server.exe)
47	VirusRAT	.NET	2013	Detected (Server.exe)
48	Xena	Delphi	2015	Detected (client.exe)
49	XRAT	.NET	2014	Detected (server.exe)
50	Remcos	C++	2016	Detected (remcos_agent.exe)

Based on the above results, it can be seen that filenames are not sufficient for use in distinguishing stubs for legitimate remote administration tools from stubs for remote access trojans. This insufficiency highlights the need for the hash-calculating aspect of this research project.

For the purposes of 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 in the table below:

Table 14

Results of the Hash Check on Tested Samples

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

[a] (ANYRUN, 2020)

Figure 10

VirusTotal Hash Scan Result for the Legitimate Stub (VirusTotal, 2020)

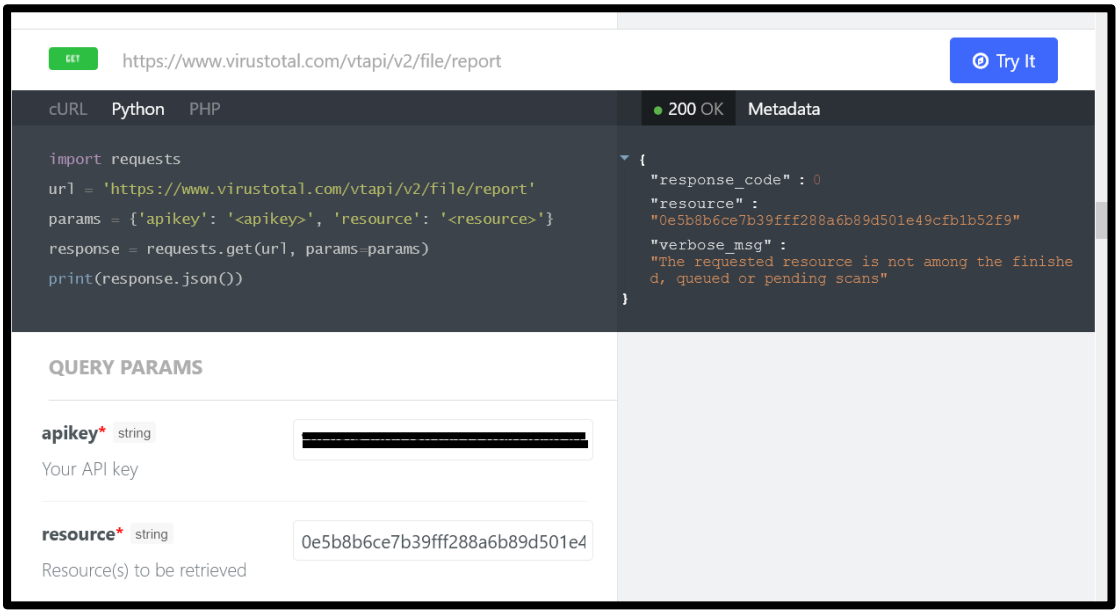
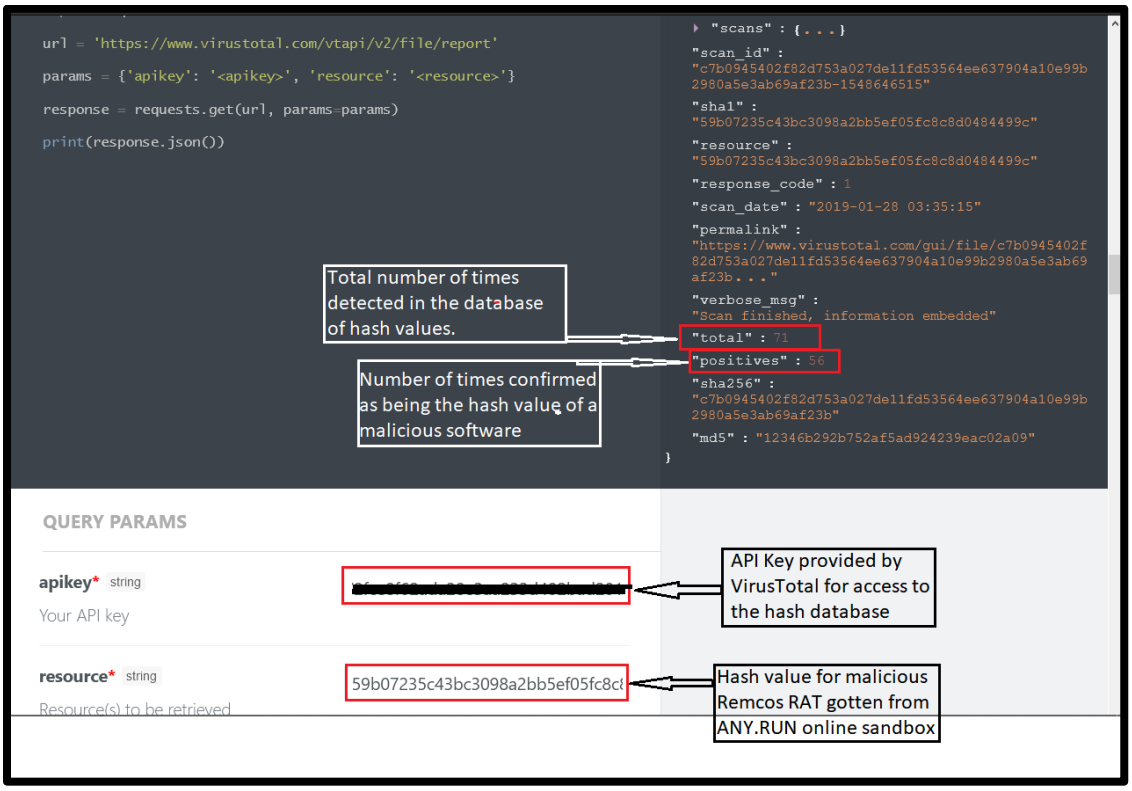


Figure 11

VirusTotal Hash Scan Result for the Malicious Stub (VirusTotal, 2020)



Discussion

Results

The results section began with a comparative analysis of some open-source RAT Detection tools as detailed in Table 11. 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 12. 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 RAT detection tool was encoded in the python programming language to make its compilation and subsequent execution non-platform specific. For the Windows platform, the source code was compiled as an "exe" file while on the Linux platform the python script will be run as it is. The source code was written on a single script to ensure as little coupling as possible within the detection mechanism. Before running the RAT detection tool's script, a separate script titled "Process List Generator.py" must be run to generate the "Process-Hash List.csv" file—one of the System Administrator Sheets. It should also be noted that the RAT detection tool's script has been written to run once and only when instantiated by the system administrator i.e. the script can be configured to run once/twice/as many times as required by the system administrator. However, this will have to be done either from inside the source code or using a batch file.

The sections of the source code functioned as follows:

- Listing of all Installed Applications: Python modules employed here were winreg & pandas.
- Process Information Gathering: Python modules employed here include psutil, datetime, time, pandas and argparse.
- Process Integrity Checking (Comparison & Hash Scans): Python modules employed here include xlrd, requests, hashlib, csv and xlsxwriter.

Table 13 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 remote access trojan 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 result is summarized in Table 14. 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 once the approved limits are exceeded. This error will come up if any of the API (Application Programming Interface) allowances below is exceeded, as provided by VirusTotal:

Table 15

API Quota

Domain	Restriction
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

Samples of each workbook, the source code, comparison scan results, RAT samples and an executable version of the source code (for Windows) can be accessed using this link.

Limitations

In the development of this research project, 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 development of this project.

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.

Finally, the lack of free actual samples of malicious RATs hindered the research from being done on live samples. This hinderance 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.

Future Research

The lack of free actual samples of malicious RATs hindered the research from being done on live samples. This hinderance 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.

Conclusions

This research project 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 (VirusTotal, 2020). 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 project 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.

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

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 several fields like control engineering, computer science and information systems. For the adequate addressing of such an

encompassing project, 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.

Finally, the lack of free actual samples of malicious RATs hindered the research from being done on live samples. This hinderance can be overcome by directly liaising with malware research companies like VirusTotal, AnyRun etc. for the controlled provision of malware samples for research purposes.

References

Ackerman, P. (2017). *Industrial Cybersecurity: Efficiently secure critical infrastructure systems.* Birmingham: Packt Publishing.

Alladi, T., Chamola, V., & Zeadally, S. (2020, April 1). Industrial control systems: Cyberattack trends and countermeasures. *Computer Communications, CLV*, 1-8. Retrieved May 12, 2020, from https://www.sciencedirect.com/science/article/pii/S0140366419319991?casa_token=CYzauiMnVjQAAAAA:Mn6fBwDSFQK0jRYcjcsjWyOryM0LkRe1TDHPQLHbZlfdb_JNW_52tXYw3VtWta4HCkwymsEGFF40

Ambati, S. B., & Vidyarthi, D. (2013). A Brief Study and Comparison of Open Source Intrusion Detection System Tools. *International Journal of Advanced Computational Engineering and Networking*, 26-32.

ANYRUN. (2020, November 13). *Public Submissions*. Retrieved from ANYRUN: Interactive Malware Analysis: https://app.any.run/submissions/#tag:remcos

Arampatzis, A. (2019, September 10). *What is the ISA/IEC 62443 Framework?* Retrieved from Tripwire: https://www.tripwire.com/state-of-security/regulatory-compliance/isa-iec-62443-framework/

Awad, A. A., Sayed, S. G., & Salem, S. A. (2019). Collaborative Framework for Early Detection of RAT-Bots Attacks. *IEEE Access, 7*, 71780-71790. Retrieved October 7, 2020

Babu, B., Ijyas, T., P., M., & Varghese, J. (2017). Security Issues in SCADA based Industrial Control Systems. *2nd International Conference on Anti-Cyber Crimes (ICACC)* (pp. 47-51). Abha: IEEE.

Cai, N., Wang, J., & Yu, X. (2008). SCADA System Security: Complexity, History and New Developments. *IEEE International Conference on Industrial Informatics* (pp. 569-574). Daejeon: IEEE .

Center for Strategic & International Studies. (2020, April). *Significant cyber incidents.* Retrieved from Center for Strategic & International Studies: https://csis-prod.s3.amazonaws.com/s3fs-public/200504_Cyber_Attacks.pdf?0M9lsdHhOXhw.BahGk5hYXgDtEWrs6x9

Cisco. (2020, June 11). *About: Features*. Retrieved from ClamavNet: https://www.clamav.net/about#overview

Cisco. (2020, September 24). *SNORT Users Manual* . Retrieved from SNORT: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node3.html

Cowan, J. (2015, March 12). *Energy sector tops list of US industries under cyber attack, says Homeland Security report*. Retrieved from IoT Now: 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/

Fan, X., Fan, K., Wang, Y., & Zhou, R. (2015). Overview of Cyber-Security of Industrial Control System. *International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC)* (pp. 1-7). Shanghai: IEEE.

Foreman, J. C., Graham, J. H., Hieb, J. L., & Ragade, R. K. (2012). *A Curriculum Model for Industrial Control Systems Cyber-Security with Sample Modules.* West Lafayette: ISCA.

Gatlan, S. (2020, February 27). *As Coronavirus Spreads, so does COVID-19 Themed Malware*. Retrieved from Bleeping Computer: https://www.bleepingcomputer.com/news/security/as-coronavirus-spreads-so-does-covid-19-themed-malware/

Graham, J., Jeffrey, H., & John, N. (2016). Improving Cybersecurity for Industrial Control Systems. *IEEE 25th International Symposium on Industrial Electronics* (pp. 618-623). Santa Clara: IEEE.

Gregg, M., & Johnson, R. (2018). *Certified Information Systems Auditor (CISA) Cert Guide.* Indianapolis, Indiana, United States of America: Pearson Education, Inc. Retrieved June 2, 2020

Haugwitz, H. v. (2019, May 19). *AIDE*. Retrieved from AIDE: https://aide.github.io/

Idrissi, O. E., Mezrioui, A., & Belmekki, A. (2019). Cyber Security challenges and Issues of Industrial Control Systems - Some Security Recommendations. *IEEE International Smart Cities Conference (ISC2)* (pp. 330-335). Casablanca: IEEE.

Imam, F. (2019, December 3). *Malware spotlight: What is a Remote Access Trojan (RAT)?* Retrieved from InfosecInstitute: https://resources.infosecinstitute.com/malware-spotlight-what-is-a-remote-access-trojan-rat/#gref

International Society of Automation. (2018). *ANSI/ISA-62443-2-4-2018 / IEC 62443-2-4:2015+AMD1:2017 CSV, Security for industrial automation and control systems, Part 2-4: Security program requirements for IACS service providers (IEC 62443-2-4:2015+AMD1:2017 CSV, IDT*. Retrieved from International Society of Automation: https://www.isa.org/store/ansi/isa-62443-2-4-2018-/-iec-62443-2-42015amd12017-csv,-security-for-industrial-automation-and-control-systems,-part-2-4-security-program-requirements-for-iacs-service-providers-iec-62443-2-42015amd12017-csv,-idt/62740948

International Society of Automation. (2018). *ANSI/ISA-62443-4-1-2018, Security for industrial automation and control systems Part 4-1: Product security development life-cycle requirements*. Retrieved from International Society of Automation: https://www.isa.org/store/ansi/isa-62443-4-1-2018,-security-for-industrial-automation-and-control-systems-part-4-1-product-security-development-life-cycle-requirements/60908278

Jiang, D., & Omote, K. (2015). An Approach to Detect Remote Access Trojan in the Early Stage of Communication. *29th International Conference on Advanced Information Networking and Applications* (pp. 706-713). Gwangiu, South Korea: IEEE.

Kim, D.-Y. (2013). Cyber Security Issues Imposed on Nuclear power plants. *Annals of Nuclear Energy*, 141-143.

Knowles, W., Prince, D., Hutchison, D., Disso, J. F., & Jones, K. (2014). A Survey of Cyber Security Management in Industrial Control Systems. *International Journal of Critical Infrastructure Protection*.

Luiijf, E. (2016). Threats in Industrial Control Systems. In E. J. Colbert, A. Kott, E. J. Colbert, & A. Kott (Eds.), *Cyber-security of SCADA and Other Industrial Control Systems* (pp. 69-93). Cham: Springer. Retrieved June 9, 2020

Maddison, J. (2018, June 21). *Resolving the Challenges of IT-OT Convergence*. Retrieved from CSO from IDG: https://www.csoonline.com/article/3283238/resolving-the-challenges-of-it-ot-convergence.html

McLaughlin, S., & Zonouz, S. (2014). Controller-Aware False Data Injection Against Programmable Logic Controllers. *IEEE International Conference on Smart Grid Communications* (pp. 848-853). Venice: IEEE.

Menze, T. (2019). *The state of industrial cybersecurity*. Dusseldorf: ARC Advisory Group GmbH & Co. KG. Retrieved May 12, 2020, from https://ics.kaspersky.com/media/2019_Kaspersky_ARC_ICS_report.pdf

Mimura, M., Otsubo, Y., & Tanaka, H. (2016). Evaluation of a Brute Forcing Tool that Extracts the RAT from a Malicious Document File. *11th Asia Joint Conference on Information Security* (pp. 147-154). Fukuoka: IEEE.

Mimura, M., Otsubo, Y., Tanaka, H., & Tanaka, H. (2017). A Practical Experiment of the HTTP-Based RAT Detection Method in Proxy Server Logs. *12th Asia Joint Conference on Information Security (AsiaJCIS)* (pp. 31-37). Seoul: IEEE.

Montalbano, E. (2020, March 6). *Spread of Coronavirus-Themed Cyberattacks Persists with New Attacks*. Retrieved from Threatpost: https://threatpost.com/coronavirus-themed-cyberattacks-persists/153493/

Obregon, L. (2015, September 23). *Reading Room.* Retrieved from SANS Institute: https://www.sans.org/reading-room/whitepapers/ICS/secure-architecture-industrial-control-systems-36327

Office of the Press Secretary. (2003). *Homeland Security Presidential Directive / HSPD-7.* Washington: Office of the Press Secretary, The White House.

OSSEC Project Team. (2020, September 25). *About OSSEC HIDS.* Retrieved from OSSEC: https://www.ossec.net/about/

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

Panayotidis, G.-N. (2019, July 5). *Organizational Policies & Procedures for ICS & SCADA Systems: Overview & Examples*. Retrieved from Study.com:

https://study.com/academy/lesson/organizational-policies-procedures-for-ics-scada-systems-overview-examples.html

Quadrant Information Security. (2020, September 25). *The Sagan Log Analysis Engine*. Retrieved from Quadrant Information Security: https://quadrantsec.com/sagan_log_analysis_engine/

Quanterion Solutions Incorporated. (2015, February 24). *NIST SP 800-82 Revision 2, Guide to Industrial Control Systems (ICS) Security*. Retrieved from Cyber Security & Information Systems Information Analysis Center: https://www.csiac.org/reference-doc/nist-sp-800-82-revision-2-guide-to-industrial-control-systems-ics-security/

Rockwell Automation. (2008). *Ethernet-to-the-Factory 1.2 Design and Implementation Guide.* San Jose: Cisco Systems.

Samhain Labs. (2020, September 24). *Samhain*. Retrieved from Samhain: https://www.la-samhna.de/samhain/

Samson, R. (2020, July 28). *The Top 10 Intrusion Detection and Prevention Systems*. Retrieved from ClearNetwork: https://www.clearnetwork.com/top-intrusion-detection-and-prevention-systems/#

Seals, T. (2018, September 7). *ThreatList: Attacks on Industrial Control Systems on the Rise*. Retrieved from threatpost: 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.

Security Onion Solutions. (2020, September 25). *About Security Onion*. Retrieved from Security Onion: https://securityonion.net/

Shang, L. (2010). The mEasuring Principle and MEthod of Telemetering Module of the Remote Terminal Unit. *International Conference on Electrical and Control Engineering* (pp. 4433-4435). Wuhan: IEEE.

SolarWinds Worldwide. (2020, February 7). *What Is RAT? Best Remote Access Trojan Detect Tools*. Retrieved from DNSstuff: https://www.dnsstuff.com/remote-access-trojan-rat

Stouffer, K., Pillitteri, V., Lightman, S., Abrams, M., & Hahn, A. (2015). Guide to Industrial Control Systems (ICS) Security. *NIST Special Publication, 800*(82), 16-16.

Sullivan, D., Luiijf, E., & Colbert, E. J. (2016). Components of Industrial Control Systems. In E. J. Colbert, & A. Kott, *Cyber-security of SCADA and Other Industrial Control Systems* (pp. 15-28). Switzerland: Springer International Publishing.

Technology, N. I. (2018). *Framework for Improving Critical Infrastructure Cybersecurity.* Gaithersburg: National Institute of Standards and Technology.

Testa, D., Lepore, M. F., Pirozzi, A., & Mella, L. (2020, February 2). *New Cyber Attack Campaign Leverages the COVID-19 Infodemic*. Retrieved from YOROI: https://yoroi.company/research/new-cyber-attack-campaign-leverages-the-covid-19-infodemic/

The International Society of Automation. (2018, October 12). *New ISA/IEC 62443 standard specifies security capabilities for control system components*. Retrieved from The International Society of Automation: https://www.isa.org/intech/201810standards/

The International Society of Automation. (2019, February 28). *United Nations commission to integrate ISA/IEC 62443 into Cybersecurity Regulatory Framework.* Retrieved from The International Society of Automation: https://www.isa.org/intech/201902standards01/

The White House Office. (2013, February 2). *Presidential Policy Directive 21: Critical Infrastructure Security and Resilience.* Retrieved from Homeland Security Digital Library: https://www.hsdl.org/?view&did=731087

Trend Micro Incorporated. (2020, June 9). *Industrial Control System*. Retrieved from Trend Micro USA: https://www.trendmicro.com/vinfo/us/security/definition/industrial-control-system

VirusTotal. (2020, November 21). *API Responses*. Retrieved from VirusTotal: https://developers.virustotal.com/reference#file-report

VirusTotal. (2020, November 4). *Yara: The pattern matching swiss knife for malware researchers (and everyone else)*. Retrieved from VirusTotal: https://virustotal.github.io/yara/

Wicked Problem Perspectives Working Group. (2020, June 11). *Strategic Standards Management*. Retrieved from Northwestern University:

https://www.northwestern.edu/standards-management/collaborators/organizations/nist.html

Wu, S., Liu, S., Lin, W., Zhao, X., & Chen, S. (2017). Detecting Remote Access Trojans through External Control at Area Network Borders. *ACM/IEEE Symposium on Architectures for Networking and Communications*, 131-141.

Zhu, H., Wu, Z., Tian, J., Tian, Z., Qiao, H., Li, X., & Chen, S. (2019). A Network Behavior Analysis Method to Detect Reverse Remote Access Trojan. *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)* (pp. 1007-1010). Beijing, China: IEEE.