Empirical Evaluation of Forensics Solutions for Live Forensics on SCADA Mchinery Control Systems

Avinash Srinivasan
Computer and Information Sciences
Temple University
Philadelphia, PA 19121
Email: avinash@temple.edu

Frank Ferrese
Naval Surface Warfare Center
Philadelphia, PA 19121
Email: frank.ferrese@navy.mil

***Abstract*—Critical infrastructures, especially machinery control systems (MCS) found onboard modern U.S. Navy warships and other military systems, are affected because of their use of commercial automation solutions. Therefore, a critical requirement is building tools and techniques to support rapid, effective, and effiicient incident response and cyberforensics capabilities. MCS being mission critical, have zero-tolerance when it comes to downtime. Consequently, a forensic investigator cannot shutdown the system to capture evidentiary data. Furthermore, todays sophisticated cyber attacks are primarily executed from the memory with little evidence on the persisten storage. Therefore, live forensics seems to be the only viable solution for conducting incident response and cyberforensics since evidentary data is primarily volatile.**

In this paper, we evaluate some of the most prpolar and powerful forensics tools and frameworks to determine their suitability for porting to mission critical MCS environment. We design and develop a rudimentary prototype of a live forensics framework for SCADA-MCS that can readily integrate the existing tools and techniques on to a common platform. Our prototype is designed conforming to *Open Systems Architecture* (OSA) to support cross vendor and cross platform solutions. Our empirical analysis highlights the strengths and weakness of tools studied proposes other requisite capabilities for tools to support a comprahensive live forensics on SCADA-MCS.

***Keywords*—Confidentiality, cyber-physical systems, data leak, fieldbus, integrity, live forensics, networks, SCADA, security.**

I. Introduction

Vulnerabilities of such MCSs with majority proprietary system components and protocols can have vulnerabilities ranging from the very basic issues such as systems without passwords or with hard-coded passwords known to everyone to configuration issues and other software bugs.

Control systems domain has slowly yet steadily transitioned from the traditional all-analog domain to a more cyber-driven domain – popularly known as the Cyber Physical Systems (CPS). Despite a rapid adoption of the new cyber-driven technology, the physical realm of the CPS is predominantly comprised of legacy and proprietary hardware. Consequently, such proprietary aspects of control systems environments present new challenges hindering the translation and integration of modern state-of-the-art forensics analysis tools and techniques into the control systems domain.

Supervisory Control and Data Acquisition (SCADA) systems are a type of Industrial Control System (ICS) that monitor and/or control legacy systems remotely. SCADA systems typically operate on large-scale processes across multiple sites, and can be used over long distances. A SCADA system consists of one supervisory system and many slave systems. Slave systems are divided into two subcategories – *Remote Terminal Units* (RTUs), and *Programmable Logic Controllers (PLCs)*.

Originally, RTU devices only collected telemetry data, and only PLC devices possessed process control capabilities. However, current incarnations of the two products have gained enough features that the distinction is no longer meaningful. When SCADA systems were first developed, they communicated solely using the Modbus serial standard, RS485. Recently, many SCADA systems have been configured to communicate via TCP/IP instead of Modbus. This allows easier integration with existing high-bandwidth infrastructure, but introduces an array of risks of exposing system vulnerabilities to a third party and the *Internet* at large.

SCADA-MCS systems were originally created to be deployed in non-networked environments. Therefore they lack of adequate security against Internet-based threats and cyber-related forensics. In recent years, SCADA systems have undergone significant technology advancement that potentially increase the risks to which they are exposed. Among these risks, one that is the most evident and crucial is its increased connectivity that could permit remote controls over the Internet, or the incorporation of general purpose tools, thus incorporating already known vulnerabilities of these.

Any cyber-attack against SCADA systems demands forensic investigation to understand the cause and effects of the intrusion or disruption on such systems. Therefore, SCADA system forensics is an essential process within the cyber-security lifecycle that not only helps to identify the cause of an incident and those responsible but to help develop and design more secure CPS and critical infrastructure systems of the future. However, a SCADA system has a critical requirement of being continuously operational and therefore a forensic investigator cannot turn off the SCADA system for data acquisition and analysis. Current live forensic tools are typically used to acquire and examine memory from computers running either Windows or Unix. This makes them incompatible with embedded devices found on SCADA systems that have their own bespoke operating system. Additionaly, only a limited number of forensics tools have been developed for SCADA systems, with no development of tools to acquire the program code from PLCs. With direct access to the memory addresses

Incident	Year	Vulnerability	Malware	Attack Vector
Ukrainian Power Outage	2015	Excel Macros	BlackEnergy	Spear Phishing
New York Dam	2013	Command & Control	N/A	Cellular Modem
European Energy Company	2016	Windws Antivirus	SFG malware	Security appliance bypass backdoor on Windows

Table I. MAJOR CYBERQATTACKS ON CONTROL SYSTEMS

of the PLC, PLC debugging tools have promising functionalities as a forensic tool , such as the "Snapshot" function that allows users to directly take values from the memory addresses of the PLC, without vendor specific software.

A. Attacks on Control Systems – Scope of the Problem

- Learning the vulnerabilities of the ICS and MCS systems and having the ability to test for possible defenses using an authentic testbed is valuable to cyber security. In particular, this line of research is in direct support of Presidential Policy Directive PPD- 21 and Presidential Decision Directive 63 to secure critical infrastructure from cyber attack [1].
- According to NCCIS/ICS-CERT [2], the number of ICS incident reported in the United States between 2012 - 2015 are as follows – 197 incidents (2012), 257 incidents (2013), 245 incidents (2014), 295 incidents in (2015).
- In July 2014, ICS customers seeking a PLC software update inadvertently downloaded a maliciously planted Trojan, dubbed Havex [8]. The Havex Trojan scans a widely used PLC standard called open protocol communications (OPC) and is believed to gather information about vulnerabilities in the target system, as a way to test code before a larger attack. The Havex Trojan affected a German manufacturing company, a Belgium PLC VPN software company and a Swiss company that produces industrial cameras.
- Another 2014 cyber-attack known as the "The Mask" [5] was a widespread espionage malware attack targeted at government agencies, energy, oil and gas companies and research organizations. The malware intercepts all communication channels and collects vital information from the victim's machine, to facilitate the theft of private information from ICS's.
- In 2014, Maritime ships using global positioning systems (GPS), automatic identification systems (AIS), and electronic chart display and information system (ECDIS) were found to have uncovered serious flaws in security [13].

B. Solution Novelty and Summary of Contributions

The proposed solution is designed conforming to the guidelines of OSA/OSE standards. Consequently, the solution framework provides several benefits over custom and proprietary solutions. Some of the the key benefits are delineated below –

- Empirically measures the performance of state-of-the-art forensics tools.

Table II. LIST OF ACRONYMS AND NOTATIONS USED.

Notation	Description
ADU	Application Data Unit
API	Application Programming Interface
ARM	Advanced Risk Machine
CPS	Cyber Physical System
DLL	Dynamically Link Library
IR	Incident Response
LKM	Loadable Kernel Module
MCS	Machinery Control System
MTU	Master Terminal Unit
OE	Operating Environment
OSA	Open Systems Architecture
OSE	Open Systems Environment
PDU	Protocol Data Unit
PLC	Programmable Logic Controller
PPC	Power PC
RAM	Random Access Memory
RMFF	Rekall Memory Forensics Framework
RTU	Remote Terminal Unit
TSK	The Sleuth Kit
VMFF	Volatility Memory Forensics Framework

- Empirical validation is performed in realworld SCADA-MCS settings.
- Analyze the suitability of MCS components to support lightweight LKM based native live forensics solutions.
- Numerous tools are compared for their performmance under the same SCADA-MCS operating conditions.
- Analyzes the potential for combining two or more existing tools efficiently to support live forensics on SCADA-MCS.
- Provides requisite capabilities for any of these tools – either independently or in combination to support comprahensive live forensics on SCADA-MCS.

C. Roadmap

The remainder of this paper is organized as follows. In section III, we review relevant background information and related works. Then in section IV, we present the target MCS model followed by a detailed discussion on the design and implementation of the proposed prototype evaluation model in section V. We present results evaluating the proposed solution framework using the proposed metrics in section VI. Finally, we conclude this paper with concluding remarks with directions for future research in section VII.

II. RELATED WORK

A large portion of research in the domain of control systems forensics has focused Programmable logic controller (PLC) firmware, which provides a software-driven interface between system inputs and physically manifested outputs, is readily open to modification at the user level. PLC is a special

form of microprocessor-based controller with proprietary operating system. Due to the unique architecture of PLC, traditional digital forensic tools are difficult to be applied.

In [15], authors propose a program called *Control Program Logic Change Detector* (CPLCD), which works with a set of *Detection Rules* (DRs) to detect and record undesired incidents on interfering normal operations of PLCs. In order to prove the feasibility of their CPLCD solution, authors set up two experiments for detecting two common PLC attacks.and illustrate the advantages of using a network traffic analyzer such as wireshark in tandem with CPLCD for performing digital forensic investigation on PLCs.

In [4] authors note that current efforts to protect PLCs against firmware attacks are hindered by a lack of prerequisite research regarding details of attack development and implementation. In their effort, authors address threats posed by PLC firmware counterfeiting and the feasibility of such attacks, this research explores the vulnerability of common controllers to intentional firmware modifications. In particular, this work derives the firmware update validation method used for the Allen-Bradley ControlLogix PLC. Through a proof-of-concept, authors demonstrate how to alter a legitimate firmware update and successfully upload it to a ControlLogix L61. Subsequently, they present possible mitigation strategies which includes digitally signed and encrypted firmware as well as preemptive and post-mortem analysis methods to provide protection. Authors conclude their work noting that their work will motivate future research in PLC firmware security through direct example of firmware counterfeiting.

In [11] authors create a high level software application capable of detecting critical situations like abnormal changes of sensor reads, illegal penetrations, failures, physical memory content and abnormal traffic over the communication channel in a SCADA environment. A key challenges authors are faced is to minimize the impact of the tool on the SCADA resources, during the data acquisition process.

Authors in [7] provide an overall forensic taxonomy of the SCADA system incident response model. The proposed taxanomy model discusses the development of forensic readiness within SCADA system investigations, including the challenges faced by the SCADA forensic investigator and suggests ways in which the process may be improved.

In [14], authors explore the problem of acquiring program code from PLCs under two hypotheses – *1) Program code is an important forensic artefact that can be used to determine an attacker's intentions;* and *2) PLC debugging tools can be used for forensics to facilitate the acquisition and analysis of the program code from PLCs.* Based on the experiments conducted using tools from NIST CFTT, the results rejected the second hypothesis since their tool – PLC Logger – had failed half of the tests. With this, authors conclude that the PLC Logger in its current state has limited suitability as a forensic tool. To be accepted as a forensically sound tool, the shortcomings have to be addressed.

In [3],

In [10],

In [12],

III. Preliminaries

MCS Vulnerabilities can be broadly categorized into the following two groups – *Platform Specific Vulnerabilities* and *Network Specific Vulnerabilities*.

A. Live Forensics

Live forensics remedies some of the problems introduced by traditional forensic acquisition by performing analysis on a live system. Live forensics evolved as a sub-discipline in response to the shortcomings of dead forensic acquisition. A key feature of live forensics is its ability to retrieve volatile information which would be absent from static or deadbox forensics. It is also useful in limiting acquired data to information relevant to the incident under investigation. The objective of both static and live forensics is the same – to find evidentiary data in support of the hypothesis. Consequently, the only differentiator is that the artifacts are being discovered on a live running system against an active adversary in case of of live forensics[1].

Live forensics also supports analysis of processes, events, actions, and the likes that are questionable and potential early indicators of an impending attack. Therefore, with appropriate live forensics tools and techniques, the analyst remotely examine running processes, dump physical memory, examine running services, etc. and determine if the situation warrants an in-depth analysis of that system. Unlike dead box forensics wherein the analyst has to be on site to acquire the evidence drives, live forensics can be performed remotely, which is not only convenient but also saves a lot of time due to false positives.

B. Control Systems

There are two primary types of control systems – *Distributed Control Systems* (DCS) and *SCADA* systems. DCS systems are typically used within a single processing or generating plant or over a small geographic area, where as SCADA systems are intended for use over larger, geographically dispersed distribution operations. The MCS used in the U.S. Navy suits DCS better than SCADA, although either can be used to remotely actuate, control, and monitor Ship's machinery equipment such as engines, electrical plants, navigation and anchoring systems, and other auxiliary systems [9].

In [6], a security testbed is proposed primarily for testing Navy's MCS for vulnerabilities.

The U.S. Navy MCS combines many systems into one network to both operate and monitor the entire engineering plant utilizing different communication protocols but does not span a large geographical area. Because of the small geographical area covered, by definition the MCS?s of the U.S. Navy are not SCADA systems. However, most recently amongst the cyber security and naval systems engineering personnel the term SCADA has become synonymous with both DCS and MCS type systems [9].

[1] $https://www.symantec.com/connect/blogs/live-response-vs-traditional-forensics-0$

C. Fieldbus – A Review

Fieldbus is the name of a family of industrial computer network protocols used for real-time distributed control, standardized as IEC 61158[2]. Fieldbus works on a network structure which typically allows daisy-chain, star, ring, branch, and tree network topologies.

1) PROFIBUS – PROcess FIeld BUS Protocol: Process Field Bus – PROFIBUS – is a standard for fieldbus communication in automation technology and used by Siemens. PROFIBUS is standardized in IEC 61158 and is the world's most successful fieldbus with 53.7 million devices installed by the end of 2015. Utilizing a single, standardized, application-independent communication protocol, PROFIBUS supports fieldbus solutions both in factory and process automation as well as in motion control and safety-related tasks. The fieldbus protocol that Siemens and other manufacturers in Europe developed evolved into present day PROFIBUS protocol.

D. The Modbus Protocol – A Review

Modbus is a protocol for serial communication using either the RS-232 standard or the RS-485 standard. Modbus allows for one master device to communicate with many slave devices. Since only one slave device can communicate with the master device at any given moment, all communication must be initiated by the master device. In SCADA systems, this means that the RTUs and PLCs (slave devices) operate mostly autonomously, except when the supervisory system (master device) issues a command or requests telemetry data.

Modbus Packet Structure The Modbus TCP protocol uses a data structure called the Application Data Unit (ADU) to transmit data at the Application Layer. Each Modbus TCP ADU is composed of a Modbus Application (MBAP) header followed by a Modbus TCP Protocol Data Unit (PDU). The MBAP header contains five data fields for a total size of seven bytes. The first field is a two-byte transaction ID, followed by a two-byte protocol ID field.

The third data field is a two-byte message length field which specifies the Modbus PDU size in bytes, and the MBAP header ends with a one-byte Unit ID field. The Modbus TCP PDU contains two fields for a total size varying between 4 - 65535 bytes. The first PDU entry is a *one-byte function code* that represents the function of the request or response, followed by a *variable length data field*. It is the data field that holds the actual payload for the corresponding request/response message.

E. MCS Operational Environment Challenges and Limitations

Below is a list of challenges and constraints in designing and developing tools and techniques to support live forensics in SCADA-MCS environment.

1) Hardware Platform Limitations: The three most popular HW ARCs for embedded components of MCS such as RTU and PLC are "ARM", "PPC", and "$x86/x64$". However, $x86/x64$ is out numbered by ARM and PPC by a significant margin. While open-source memory forensics frameworks such a Volatility and Linux do provide support for ARM and PPC, they are primarily built around $x86/x64$ systems. This is due to the enormous market share of Desktops/PCs.

2) Software Platform Limitations: Majority of networked components for SCADA run on legacy platforms with extremely small quantities of physical memory. Consequently, they have little room for natively expanding their capabilities to support augmented logging for IR and forensics analysis.

3) Lack of Standards and API Definitions: There is very limited information about the SCADA MCS architecture and system APIs available for building enhanced plug-in tools to support out-of-band processing to support forensics.

4) Proprietary Solutions: Proprietary tools have limited capabilities and economically infeasible. Furthermore, proprietary tools often tend to be tailored for very specific tasks and platforms. Such lack of flexibility is very concerning given the evolving and volatile state of cyber-attacks. Existing proprietary and open-source tools with live forensics capabilities are prohibitive from a resource standpoint – massive memory footprint and significant IO bound operations topping the list of resource concerns.

5) Ready Adoption Challenges: Adoption of custom open-source tools "as-is" can present significant challenges with regards to legal and regulatory requirements. Porting necessary requirements for cyberforensics of SCADA system components into existing open-source tools warrants significant effort.

6) High Degree of Proprietary Solutions and Heterogeneity: Not only are the solutions proprietary in nature due to the unique domain requirement of each and every control systems environment, but also there degree of heterogeneity among field devices even within the same domain is significant to have a "one-size-fits-all" solution. Such proprietary solutions with such heterogeneity will make live forensics extremely challenging some of which are as follows – network constraints due to OEM restrictions, node/device constraints due to variations in HW/SW, communications security due to legacy and resource constrained devices and proprietary protocols, and the likes.

State-of-the-art tools with live forensics capabilities primarily include *Volatility* and *Rekall*. Both this frameworks currently provide very limited support for ARM and PowerPC (PPC) memory architectures, since they have been primarily developed around the x86 systems. While it is possible to expand either of these frameworks to support live forensics on MCS systems, it is significant effort to write a plugin/LKM for ARM/PPC memory architectures taking into considerations the subtle variations implemented by vendors creating proprietary memory architecture.

IV. Prototype Model

We utilize two vastly different methods – Memory Analysis and Traffic Analysis. There two approaches have different purposes with regards to security and forensics analysis. However, the two methods can be coulped to work complimenting each other. Such a hybrid platform will enable robust security and forensics analysis. The open systems architecture prototype framework will use the following open source tools along with numerous custom tools.

[2] $https://en.wikipedia.org/wiki/Fieldbus$

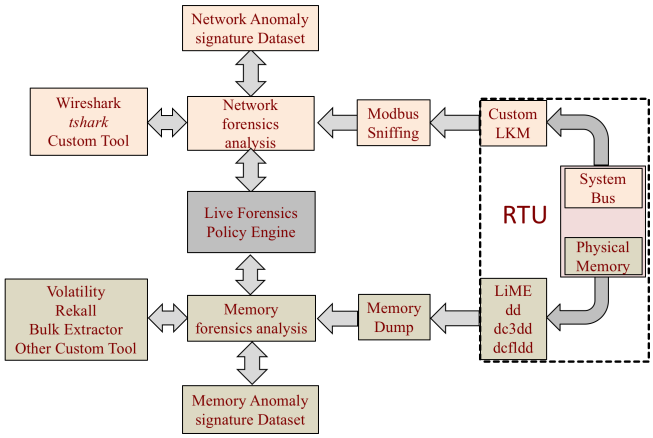

Figure 1. Open Systems Architecture prototype supporting memory dump and fieldbus traffic analysis.

- **LiME (Linux Memory Extractor).** LiME is an open source loadable kernel module that can acquire live memory from popular processor architectures – x86/x64, PPC, and ARM. Furthermore, LiME can extract the live memory in three different formats – .lime, .raw, . Compatible with desktop Linux distributions and Android. Does not include any tools to analyze the images it collects.
- **Bulk Extractor.** This is an open source file system forensics analysis tool suite. Unlike, LiME, bulk extractor is not capable of acquiring live memory. Furthermore, bulk extractor is also not capable of extracting process level information from a memory dump. However, the tool can scan memory dump images for certain types of forensically relevant information such as – IP address, Network sockets, Credit card numbers, URLs, Passwords, and Encryption keys. While the tool evidently has limitations in processing live memory, its limited capabilities are very useful in correlating live memory information and raw packet capture providing a unique purview during security and forensics analysis.

V. Validation Model

A. Validation Framework

We conduct some testing on Rekall's WinPMem tool to determine how different operational parameters affect the time and storage space it requires to capture memory dumps. Our tests included:

1) Capturing memory dumps of the same system with different amounts of system memory installed, and recording the time and storage overhead for each.
2) Capturing memory dumps using different file compression options: the snappy algorithm, the zlib algorithm, and no compression. Although Rekall?s website claims that snappy compression is the fastest option, we found that storing the files in uncompressed form was often slightly faster on the machine we used.
3) Observing WinPMem's behavior when we try to append a new memory dump to an existing one. This is the default behavior for when the specified output file already exists.

B. Transmitting memory dumps via Modbus TCP

From number of bytes (2) in the message length field, we conclude that each Modbus TCP frame can hold a payload up to 65535 bytes in size. Since very few embedded systems with system memory of 64 KB or less are still in use, any implementation of memory dump transmission via Modbus TCP should be designed to accommodate file splitting on the server side and file splicing on the client side. Table 1 shows how different sizes of memory dump would be split for transmission:

1) Packet Crafting Module: We have implemented a custom kernel module that is a packet crafter such that the memory dump can be packaged into into independent data packets and/or embedded into unused bits of data packets exchanged between the field device and the MTU over the course of several request/response messages.

VI. Validation Results

A. Observations

- Winpmem provided by Rekall is expected to either append a complete memory dump to the existing one and double its size, or to append a subset of the address space that had changed since the acquisition of the previous dump. Instead, the program returned an error. While we have every intention to investigate this further, for the work presented paper, we dont have additional information.
- Rekall has a small but powerful suite of built-in plugins to analyze Windows registry entries. In this example, we will use them to find and display registry key pairs contained in our memory dump. With our image file open in Rekall, we begin by calling the hives plugin.

B. Results

C. Secure Memory Dump Transfer to MTU

Protecting the data against corruption and truncation would require additional measures. Two possibilities exist to alert the client to any errors in data transmission or splicing:

1) Transmit a single checksum for the entire memory dump, and verify the integrity of the reassembled dump-file at the MTU.This would require very little overhead. The comparison in Table **??** shows that for several common RAM configurations, packaging a checksum from the Linux command sha256sum with the memory dump requires no extra packets to be sent. Only one checksum of the memory dump file would be necessary because TCP already performs checksum-based verification of each packet. The only concerning aspect of this method is its potential to become too complex, in terms of computation and disk access, to be feasible on an embedded system.

Open Source Tools	Capabilities	Limitations
Rekall Memory Forensics Framework	A computer forensics tool that scans a memory dump image or virtual machine snapshot and extracts useful information	Requires a "profile" (aka config file) to analyze images of systems with address spaces that aren't included in its releases
	Includes a toolkit for acquiring memory dumps on Windows, Linux, and OSX systems	Generating profiles requires compiling Rekall provided kernel module
	Can be used either as a standalone application or as a scripting library to automate memory analysis	Analysis capabilities are primarily limited to x86/amd64 systems
Volatility Memory Forensics Framework	Functionality can be extended with plugins and address space profiles	Volatility is not capable of acquiring memory dumps – it is dependent on other tools like LiME
	Forensics tool-suite that scans and extracts information from memory dump images of Win, OSX, Linux, & Android systems	Framework's profile subsystem has compatibility limitations similar to that of Rekall
		Compatible with ARM systems, primarily Android, does not carry over to other embedded ARM systems running desktop Linux
Volatilitux	Released in 2010 to add Linux compatibility in Volatility	Not actively updated since 2010, and has limited support for Linux Kernel beyond 2.6.x
	Has compatibility with ARM-based desktop Linux distributions	Compatibile only with images captured using Volatilitux
		Very limited analysis capability – only 5 commands available
LiME (Linux Memory Extractor)	Linux LKM capable of acquiring memory dump in several different formats – ".lime", ".raw", and "padded"	Lacks memory dump processing and analysis capability
	Compatible with desktop Linux distributions and Android	
Dumpchk.exe	Microsoft freeware for validating Windows memory dumps	Only supports Windows memory dumps
	Limited capability of parsing certain types of metadata from memory dumps	Proprietary and non-extensible
Autopsy/ The Sleuth Kit (TSK)	Disk forensics analysis tool capable of handling numerous file system types for disk analysis	Does not support memory acquisition or memory analysis
Bulk Extractor	Disk forensics analysis tool	Can only process acquired memory images for network connection, registry keys, sockets, etc.
	Capable of analyzing disk images for certain types of forensically relevant information (phone, CC numbers, URLs, etc.)	Does not support memory analysis at process level unlike Volatility or Rekall

Table III. SUMMARY OF CAPABILITIES AND LIMITATIONS OF OPEN-SOURCE FORENSICS TOOLS THAT SUPPORT LIVE FORENSICS BOTH DIRECTLY AND INDIRECTLY.

	zlib	zlib with truncation	snappy	snappy with truncation	No compression No truncation	No compression with truncation
Total runtime (s)	451.15	435.10	329.14	349.02	311.57	309.72
Size of Memdump (MiB)	5100.24	5188.26	4791.85	5646.46	8168.15	8168.15

Table IV. COMPRESSION TIME AND SIZE ON RASPBERRY PI 3.

2) Overlap the transmissions such that each fragment is transmitted with a small amount of data from the previous fragment. This method would decrease the complexity of disk and computational operations, but transmission overhead would increase proportionally to the size of the overlap.

VII. CONCLUSION AND FUTURE WORK

VIII. DEFINITIONS

Definition 1. *Cyber-Physical Systems are co-engineered interacting networks of physical and computational components that are built from, and depend upon, the seamless integration of computational algorithms and physical components.*

Definition 2. *Control Systems manage, command, direct or regulate the behaviour of other devices or systems ranging from a home a thermostat controlling a domestic boiler to large industrial control systems which are used for controlling processes or machines.*

Definition 3. *Industrial control system (ICS) is a general term that encompasses several types of control systems and associated instrumentation used in industrial production, including supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and other smaller control system configurations such as programmable logic controllers (PLC) often found in the industrial sectors and critical infrastructures.*

Definition 4. *A programmable logic controller (PLC), or programmable controller is an industrial digital computer which has been ruggedised and adapted for the control of*

manufacturing processes, such as assembly lines, or robotic devices, or any activity that requires high reliability control and ease of programming and process fault diagnosis.

Definition 5. *Supervisory control and data acquisition (SCADA) is a control system architecture that uses computers, networked data communications and graphical user interfaces for high-level process supervisory management, but uses other peripheral devices such as programmable logic controllers and discrete PID controllers to interface to the process plant or machinery.*

Definition 6. *A remote terminal unit (RTU) is a microprocessor-controlled electronic device that interfaces objects in the physical world to a distributed control system or SCADA (supervisory control and data acquisition) system by transmitting telemetry data to a master system, and by using messages from the master supervisory system to control connected objects.*

Definition 7. *A Distributed Control System (DCS) is a computerised control system for a process or plant, in which autonomous controllers are distributed throughout the system, but there is central operator supervisory control.*

Definition 8. *A loadable kernel module (LKM) is an object file that contains code to extend the running kernel (aka base kerne) of an operating system typically used to add support for new hardware (as device drivers) and/or filesystems, or for adding system calls to provide additional functionality.*

Definition 9. *Advanced RISC Machine (ARM) is a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments. ARM has reduces costs, heat and power use making it highly desirable for embedded systems in SCADA-MCS environment.*

REFERENCES

[1] Executive order 13636: improving critical infrastructure cybersecurity. June 2013.

[2] Ics-cert year in review. 2015.

[3] Irfan Ahmed, Sebastian Obermeier, Martin Naedele, and Golden G Richard III. Scada systems: Challenges for forensic investigators. *Computer*, 45(12):44–51, 2012.

[4] Zachry H Basnight. Firmware counterfeiting and modification attacks on programmable logic controllers. Technical report, DTIC Document, 2013.

[5] S. Curtis. 'the mask' cyber spying operation targets government agencies. February 2014.

[6] Nathan H Desso. *Designing a machinery control system (MCS) security testbed*. PhD thesis, Monterey, California: Naval Postgraduate School, 2014.

[7] Peter Eden, Andrew Blyth, Pete Burnap, Yulia Cherdantseva, Kevin Jones, Hugh Soulsby, and Kristan Stoddart. A forensic taxonomy of scada systems and approach to incident response. In *Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research*, pages 42–51. British Computer Society, 2015.

[8] M. Mimoso. Motives behind havex ics malware campaign remain a mystery. July 2014.

[9] Timothy Scherer and Jeffrey Cohen. The evolution of machinery control systems support at the naval ship systems engineering station. *Naval engineers journal*, 123(2):85–109, 2011.

[10] Joe Stirland, Kevin Jones, Helge Janicke, and Tina Wu. Developing cyber forensics for scada industrial control systems. In *The International Conference on Information Security and Cyber Forensics (InfoSec2014)*, pages 98–111. The Society of Digital Information and Wireless Communication, 2014.

[11] Pedro Taveras. Scada live forensics: real time data acquisition process to detect, prevent or evaluate critical situations. *European Scientific Journal, ESJ*, 9(21), 2013.

[12] Ronald M van der Knijff. Control systems/scada forensics, what's the difference? *Digital Investigation*, 11(3):160–174, 2014.

[13] J. Wagstaff. All at sea: global shipping fleet exposed to hacking threat. April 2014.

[14] Tina Wu and Jason RC Nurse. Exploring the use of plc debugging tools for digital forensic investigations on scada systems. *The Journal of Digital Forensics, Security and Law: JDFSL*, 10(4):79, 2015.

[15] Ken Yau and Kam-Pui Chow. Plc forensics based on control program logic change detection works. *The Journal of Digital Forensics, Security and Law: JDFSL*, 10(4):59, 2015.