DOI: http://dx.doi.org/10.14236/ewic/ICS2016.15

Towards a Distributed Runtime Monitor for ICS/SCADA Systems

Andrew Wain, Stephan Reiff-Marganiec
Department of Computer Science
University of Leicester
Leicester, UK
{ajbw1,srm13}@le.ac.uk

Kevin Jones
Airbus Group Innovations
Newport, UK
kevin.jones@airbus.com

Helge Janicke
Software Technology Research Laboratory
De Montfort University
Leicester, UK
heljanic@dmu.ac.uk

Industrial Control Systems (ICS) and SCADA (Supervisory Control and Data Acquisition) systems are typically used in industries such as electricity generation and supply, gas supply, logistics, manufacturing and hospitals and are considered critical national infrastructure. The evolution of these systems from isolated environments into internet connected ones, in combination with their long service life and real-time nature have raised severe security concerns in the event of a cyber-attack. In this paper, we review the current literature surrounding the threats, vulnerabilities, exploits and existing approaches to securing vulnerable SCADA systems. We then focus specifically on the development of a distributed online runtime monitor to detect violations of safety properties. We conclude with suggestions for further research needed to progress the state of the art in the area of distributed online runtime verification of SCADA systems.

SCADA, ICS, cyber, security, distributed, run-time, monitoring

1. INTRODUCTION

ICS and SCADA systems are typically used for the control of systems used in electricity generation and supply, gas supply, logistics, manufacturing and hospitals. Their uninterrupted and safe operation is critical to the safety of human lives and a nation's security. Typically these systems operate continually and have lifespans measured in decades. Whilst cyber attacks against SCADA systems are rare, occurrences are on the increase and the consequences can be severe. Let us consider just the following examples:

In 2009 a hospital security guard installed malware on hospital machines and took control of the systems controlling heating and air conditioning (Nicholson et al. 2012).

In 2010 the Stuxnet worm was discovered in Iran's power plants. The worm had been introduced on removable media and propagated from the corporate network to the SCADA network where it searched for specific models of PLC and rewrote the program logic to alter centrifuge timings to decrease their service life (Falliere et al. 2011).

In December 2015, the Ukrainian power grid suffered a cyber attack which caused 225,000 customers to lose power for several hours while the SCADA systems were manually restored (E-ISAC 2016).

Current efforts in SCADA security focus on the development of more secure architectures, message logging for offline analysis, encryption and firewall improvements. We find that current methods offer limited protection for 'semantic attacks' where the system continues to operate but in a subtly different way – as was the case with the 2010 Stuxnet attack. We therefore propose to pursue distributed online runtime monitoring as offering an additional layer of protection for vulnerable SCADA systems by inspecting and verifying states of distributed components at runtime. There are several crucial requirements associated with runtime monitoring which are particular to ICS/SCADA including the limited bandwidth available for communication thus not allowing heavy control protocols, the time critical nature of observing issues and addressing them, and the expected incomplete understanding of system configuration and implementation details.

The rest of this paper introduces the history and typical problems with SCADA systems in section 2, considers common efforts to securing SCADA systems in section 3 and then turns it is attention to where we believe progress is required: the distributed monitoring of ICS/SCADA systems

© Wain et al. Published by
BCS Learning & Development Ltd.

(Section 4). We conclude the paper with a vision of the research aspects to be addressed to achieve effective protection at runtime.

2. HISTORIC DEVELOPMENT AND TYPICAL CONCERNS

Traditional SCADA systems centred around central mainframe computer which communicated with field devices such as Programmable Logic Controllers (PLCs), Remote Terminal Units (RTUs), Intelligent Electronic Devices (IEDs) and HMI (Human Machine Interface) workstations. Local devices communicated over a serial bus, whilst remote devices communicated via telephone lines or radio technology. These early systems relied on physical security and each implementation used proprietary protocols offering 'security by obscurity' (Nicholson et al. 2012).

Through the 1980s and 1990s, SCADA architectures followed the trend of general information systems towards a distributed architecture in order to balance reliability across nodes and to take advantage the increasing computational power across the network. This led to the adoption of 'standard' network protocols such as TCP/IP to provide LAN and WAN communications with the existing proprietary protocols laid on top.

In the new millennium the overlap between traditional information systems and SCADA has continued. The introduction of open standards and off-the-shelf hardware, reduced the effect of 'security by obscurity'. Where there had once traditionally been an 'air gap' between the standard corporate network and the SCADA network, these are often connected such that the SCADA databases can be read from or cloned in the corporate network for reporting, monitoring and remote management purposes.

The increased use of standard PC hardware, operating systems and infrastructure reduced cost but opened the system to new threats as they were then exposed to the vulnerabilities common to these general computer systems (Worms, Viruses, Trojans spear phishing attacks etc.). This allows unique exploits due to the combination of the insecure nature of the proprietary protocols and the connected nature of the systems (Nicholson et al. 2012; Cárdenas et al. 2008). We explore these vulnerabilities in Section 2.3.

2.1. Threats

The threats to reliable and safe SCADA operation can originate from (1) operator or programming error from lack of training or experience, (2) malicious access to process communication channels leading to potential loss of intellectual property, (3) malicious alteration of process engineering data whilst it traverses the network or (4) flooding the network with malformed or spurious messages(Giani et al. 2009).

2.2. Attackers

Nicholson et al. (2012) highlight the following potential attackers of a SCADA system: (1) State sponsored cyber-warfare, (2) Terrorist Organisations, (3) Hackers / Organised Crime, (4) Employees / Inside attacks (5) 'Script kiddies' and Hobbyist Hackers and (6) Hacktivists.

In the case of cyber-warfare and terrorist organisations, the motivation is to cause as much disruption as possible. A state sponsored attack has almost unlimited resources and funding. Organised criminals are more likely to be motivated by the theft of insider knowledge or to include a SCADA attack in part of a larger criminal activity. This could include espionage from other companies with the motivation to give them a competitive advantage. Such an attack is likely to be more subtle and current techniques are less likely to offer quick detection.

Hackers and 'script kiddies' probably have less inside knowledge of the plant they are attacking but will be seeking known exploits and weaknesses such as default passwords and known yet unpatched vulnerabilities. They could be motivated either to cause disruption to the plant or for the personal challenge. Hacktivists could be motivated to attack critical infrastructure to draw attention to their goals and to cause disruption, for example an anti-nuclear energy protest disabling a nuclear power station.

Some attacks may be unmotivated, but instead caused by accidents as a result of lack of employee training or system faults which result disruption to normal operations of the SCADA facility.

2.3. Vulnerabilities

Due to the nature of SCADA systems, they operate continuously with infrequent maintenance windows. Test and/or development environments are not available and depending on the nature of the system there may be a strict certification process required before changes can be implemented. This means that whilst robust procedures may be established for installing patches and updates in the corporate network segment, it is very common for the SCADA segment to remain **unpatched** for a significant amount of time. (Cárdenas et al. 2008). As the complexity of the systems increases so does the occurence of bugs, and hence the need for patching. Cárdenas et al. (2008) indicate that flexible configuration options such as remote management via a web server increase the surface area for attack.

Traditional SCADA systems were physically isolated from the internet, hence physical security was considered to be adequate protection. Now that these systems which were designed to be isolated, are connected to the internet, this approach no longer offers adequate protection (Cárdenas et al. 2008; Vukovic et al. 2012; Cárdenas et al. 2008). Worse still, older devices using open protocols cannot easily be upgraded to include authentication or encryption, they must either be replaced or augmented with additional hardware. However, this would require taking the systems offline for an unacceptable period of time and be prohibitively costlyTsang Smith (2008).

SCADA systems historically used proprietary ad-hoc protocols: even the same models of equipment deployed by the same engineers could vary between implementations. This caused problems during maintenance and prompted the development of more standardised protocols such as ELCOM-90, Modbus, DNP, Fieldbus and ASCII (Kalapatapu 2004). Whilst this enables interoperation to occur between vendors of SCADA equipment, the use of open protocols also makes it easier for a would-be attacker to familiarise themselves with the system. The open design does however allow for the easier identification and correction of bugs [1], which could go undetected (but therefore unexploited) for years in a proprietary system (Cárdenas et al. 2008). Adopting the use of commodity IT extends further to hardware and operating systems, so that SCADA systems inherit the vulnerabilities of these components (see e.g. (Cárdenas et al. 2008)).

Many of the systems still in use today were designed before the strong need for encryption and due to their low computing power are unable to handle the additional complexity. Also where a SCADA system is currently in use or widely distributed, retrofitting encryption must be done in an incremental approach as it is infeasible to upgrade the entire system concurrently (Dawson et al. 2006). This leaves such systems vulnerable to eavesdropping, man-in-the-middle and replay attacks.

SCADA systems operate within strict timing constraints which must be observed to ensure safe operation. Many of these network links are running at peak load and cannot cope with the additional overhead introduced by the use of authentication headers and encryption which could effectively act as a denial-of-service attack against nodes unable to cope with the additional computation.

Rule sets used for intrusion detection work well in a standard corporate network where the attack patterns are common and are relatively easy to identify. In a SCADA environment the attacks are infrequent and likely to be unique to the specific environment under attack, therefore rule sets built from the analysis of one attack are unlikely to detect future attacks (Hadžiosmanović et al. 2014).

Existing intrusion detection systems (IDS) inspect network traffic on a corporate network which is using common protocols such as TCP/IP, SMTP, POP3, IMAP, FTP, HTTP, HTTPS etc. They do not deeply inspect the SCADA traffic and are unable to identify messages which are semantically invalid, for example the activations of the heating element in a water heater when no water is present. This kind of semantic violation was the basis for the Stuxnet attack (Janicke et al. 2015).

It is suggested by Cárdenas et al. (2008), that one of security challenges that separates SCADA security from general ICT security is that SCADA systems interact with the physical world in ways that general ICT security measures are not designed to consider. This is further aggravated by the need to locate RTUs and IEDs in remote and hostile locations (electrical substations or pumps on arctic oil pipelines for example), so physical security remains an issue, Retrofit security technologies to all these nodes concurrently while the system remains active throughout, the ability for mixed operation and to secure the infrastructure in stages is required (Dawson et al. 2006). These matters also apply to the fledgling Internet of Things (IoT) domain which will benefit from the considerations in this paper while possibly offering some low cost options for additional monitoring in SCADA if an appropriate secure monitoring solution can be found.

3. SECURING ICS/SCADA

We shall now proceed to discuss some related works from the literature with a focus on, identifying techniques for the security of ICS/SCADA systems.

3.1. New Architectures

Many new architectures have been proposed which increase the resilience and security of SCADA systems. Whilst these could be applicable to newly developed control systems, they do not address vulnerabilities in systems which will continue to be operational for decades to come. Some examples of these architectures include:

A modified version of fieldbus including a GPS receiver for time synchronisation and the use of a Hamming encoding to allow for the detection and correction of bit-level errors with a predictable correction rather than requiring the data to be retransmitted (Erdner Halang 2004).

[1] Assuming the relevant patches are deployed, see 2.3

A combination of Wireless Sensor Networks, Mobile Ad-Hoc Networks and the internet to produce a secure attack-resistant architecture (Kumar et al. (2014)) which they benchmarked against the existing techniques: Vukovic et al. (2012)'s NAMDIA (Network-Aware Mitigation of Data Integrity Attacks), Morris Pavurapu (2010)'s Retrofit IDS (Intrusion Detection System), and Fovino et al. (2012)'s CSBF (Critical State-Based Filtering).

The use of an agent-based decision support system for the automation of control decisions in large and complex control processes has been suggested in (Prayati et al. 2007). A new agent-based architecture to allow for greater configuration and adaptability on production lines where the products to be produced are customisable / change frequently Dionisio Rocha et al. (2015) follows similar premises.

The use of Component Oriented Programming (COP) as opposed to Object Oriented Programming (OOP) in large-scale and distributed SCADA systems. This has the advantage of distributing the requirements of resources and provides examples of automated discovery. However, it is not clear how this could be applied or integrated with existing infrastructures and how timing constraints could be guaranteed due to the discovery process (Anh Chau 2009).

3.2. Message Logging

Message logging can be used for offline analysis of the network and to identify traffic patterns during normal operation as well as highlighting traffic deviations from normal. Morris Pavurapu (2010), propose a data logger for serial communication based on MODBUS and DNP3 which can be retrofitted to existing applications.

Tupakula Varadharajan (2014), suggest adding a Virtual Machine based Attack Detection Agent (ADA) and Attack Detection Servers (ADS) to the infrastructure to monitor and detect anomalies in the control system. However, their monitor validates the the runtime state of the system "at random intervals", which raises the question of whether the monitor may miss unsafe states. Detection rules are initially manually configured, requiring detailed knowledge of the process under observation and they are later manually refined by log file analysis.

3.3. Encryption

Several attempts have been made to implement encryption between components of SCADA networks and also in efficient key management.

The security of unattended remote stations cannot be assumed and techniques such as Wright et al. (2004) and Tsang Smith (2008) could be thwarted bcause messages could be injected into the encrypting device (Giani et al. 2009). However, if physical access to a remote station is gained then "greater disruption and damage can be caused by other means" (Wright et al. 2004).

Key management is crucial for all encryption approaches. Beaver et al. (2002) present a cryptographic key management algorithm for SCADA systems which uses a combination of symmetric and public keys, the mechanism differs between master-controller and peer-to-peer links. Dawson et al. (2006) builds on the work of Beaver et al. (2002), by providing both a unified communication technique for master-controller and peer-to-peer connections using only symmetric keys instead of public keys in order to to reduce network traffic load. Choi et al. (2009) present a solution to the problem of broadcast communications, to be used in the event of, for example broadcasting alarm states. Lee et al. (2008) integrate Choi et al. (2009) with the Iolus framework (Mittra 1997) to break down key distribution into hierarchies to ensure efficient distribution of new keys and revocation of expired keys, whilst still supporting broadcast messages.

PiÃ¨tre-CambacÃ©dÃ¨s Sitbon (2008); Pal et al. (2009), provide overviews of the constraints and requirements for key management and current key management practices and their applicability to SCADA.

3.4. Firewall and Intrusion Detection Systems

The move to standard PCs and network hardware and connection to the corporate network opens vulnerable SCADA systems to the internet. Firewalls are generally deployed at the boundaries between the internet, corporate network and SCADA network, however they are not typically designed to inspect SCADA-specific protocols and offer limited protection for the SCADA network. There have been previous works on incorporating SCADA traffic monitoring into firewalls and IDS (Intrusion Detection Systems), such as the VIKING project by Giani et al. (2009), which suggests using an application-level module in the IDS to detect data anomalies and suspicious traffic.

Hadžiosmanović et al. (2014) used a network tap to capture and inspect raw network packets at the PLC interface. Their approach inspects the content of messages and categorised them as (1) control, (2) reporting, (3) measurement and (4) program state. Variables in program state are then categorised as (a) changing continuously, (b) changing gradually over time, (c) attribute data from a fixed set of values, or (d) never change. Statistical models are then used to ensure that messages do not deviate from their observed categorisation. This assumes that the traffic

offers a true representation of state; that messages have not been forged or manipulated and that all behaviour is observed and categorised, which may not be the case for alarm triggers and recovery operations.

Monitoring is also required at a semantic level, as otherwise ordinary messages could occur in dangerous sequences. For example, the activation of a heating element in a water heater whilst it is empty. It is therefore important to monitor in a way that is aware of the current state of multiple components throughout the system.

In the event of a Stuxnet style semantic attack, where the traffic is altered in subtle ways to affect the process under control (Janicke et al. 2015), knowledge of the content of messages is essential in order to protect the SCADA system from attack. Due to the proprietary nature of the protocols still in operation and the need to tailor the protection rules specifically for the system in question, it is unlikely that a one-size-fits-all set of firewall rules would offer the same level of protection as a monitor capable of checking the state of various components of the system with rules governing safe operations.

4. RUNTIME VERIFICATION

Existing IDS techniques are unlikely to identify a semantic attack where the messages are valid according to the SCADA protocol, but the content of the message has been altered in a way which causes the overall system to behave incorrectly, as in the case of Stuxnet. Runtime verification offers the opportunity to build a safety monitor which is capable of monitoring system states in real-time in order to detect and react to safety violations.

Runtime verification differentiates itself from formal model checking and theorem proving in that properties of a program are checked by execution instead of analysing the source or compiled code to prove safety and security properties Leucker Schallhart (2009). Runtime verification can also be applied either where no system model exists, or to complement formal model checking by verifying the implementation matches the specification. The disadvantage of runtime verification is that it can only verify *observed behaviour*, that is to say that it cannot be used to reason about the system as a whole unless every possible state occurs Leucker Schallhart (2009); Malakuti et al. (2011) during the execution. Runtime verification leads to the potential for a system to react and raise alarms or take corrective action before faults become failures.

Runtime verification can either be performed online whilst the target system is running, examining program output during execution, or offline by examining logs files and traces Leucker Schallhart (2009).

In the following we investigate the previous work in runtime verification for embedded control systems in general (4.1), SCADA systems (4.2), and distributed systems (4.3).

4.1. Runtime Verification of Embedded Control Systems

Embedded systems are constrained by low resources and computational power, with requirements for real-time and reliable operation. They are often deployed in harsh environments and must accept input from unreliable sensors yet they must provide reliable real-time operation. In order to achieve this, runtime verification has been given some attention in the context of embedded systems.

Runtime verification is presented in the context of ultra-critical embedded systems by Pike et al. (2011), who define an ultra critical system as an embedded system which senses and/or controls the physical world within fixed and time-critical constraints. Runtime verification of these systems must account for hardware faults and random failures, in addition to software design faults. The identified requirements are: (1) Functionality (the presence of RV technique cannot change the behaviour of the target system), (2) Certifiability (the RV system must not require the re-certification of the target system), (3) Timing (the RV technique must not affect the timing of the monitored system) and (4) Size, Weight, Power (the RV system must not exceed these tolerances).

These constraints prohibit the use of traditional runtime verification techniques such as instrumentation of the target system for additional monitoring output. Watterson Heffernan (2008) present a method for using the formal Java-MaC runtime verification method to monitor a Java control system using instrumentation: The control system and monitor are implemented in a JOP (Java Optimised Processor) software processor, embedded in an FPGA (field programmable gate array) and run alongside the control system

Neukirchner et al. (2012) investigate the monitoring the timing of activation patterns for tasks at runtime with verification against a timing model. Uncharacteristically long or short activation times for a task could be indicative of a fault or tampering with the system. Gu et al. (2014) combat the high-overhead of control flow checkers (CFCs) by using

partial instrumentation to ensure that the worst-case execution time (WCET) is acceptable in the embedded environment.

TyTAN is proposed by Brasser et al. (2015), as a secure architecture for low-resource embedded systems supporting dynamic loading of secure tasks, secure inter-process communication and real-time guarantees.

Despite not being SCADA specific, these techniques are of interest as the runtime monitoring of timings could be indicative of interference e.g. short delays between packets in the case of a replay attack. However, in order to thoroughly monitor the activity in the system we must be able to examine variable values for semantic correctness.

4.2. Runtime Verification of SCADA

CASPER was developed by Barbosa et al. (2012), to monitor network traffic and uses Complex Event Processing and Hidden Markov Models to predict failures in distributed safety-critical systems. The experiments monitored: (1) round trip time, (2) message rate and (3) number of requests without a reply. The Topology Detector Component builds a graph representing the network by monitoring network traffic sources and destinations. This is a useful technique in re-discovery of topology in large and complex SCADA systems, where a traditional network scanner can cause PLC failure in the form of a DOS attack.

Mao et al. (2015) built a non-intrusive monitor for the runtime behaviour of open SCADA systems. Their technique involved a Virtual Machine Monitor (VMM) to capture network traffic travelling between the PLC and the HMI. Messages are modelled and marked with guard suffixes to identify messages which cannot occur concurrently. At runtime the Sort network monitor is used to capture all IP traffic, filtering for specific packets relating to SCADA protocols. The main stages of their monitor are (1) Event Extraction, (2) Semantic Reconstruction, (3) State Refinement and (4) Behaviour Checking. They do not discuss the performance characteristics of the monitor and their model is restricted to concurrency and dependency relationships of messages, they do not appear to model or verify the message content, timeliness or source/destination of the message.

Janicke et al. (2015) developed a low-cost monitoring solution based upon an Arduino Yun and Tempura, an executable subset of Interval Temporal Logic Moszkowski (1984); Cau et al. (2009); Hale (1988). Their solution was tested using a Siemens S7-1200 PLC with the Arduino connected to the Profinet interface. The advantage of the ITL-based monitor is that ITL allows for a much richer specification of the monitor, allowing for the specification of conditions based upon the next state, sometimes true, always true, never true and upon the number of states between conditions. The solution allows inspection of the PLC registers, as represented in the network traffic, in order to identify subtle violations of safety conditions.

Tupakula Varadharajan (2014) present a virtual machine monitor (VMM) based technique to detect anomalies in network traffic and the state of virtual machines operating the HMI, Historian and current databases, and SCADA servers for anomalous behaviour. The architecture is based on the assumption that these servers are based on a virtualisation technology, which is not necessarily the case in existing SCADA systems especially in strict real-time environments.

TAIGA (Franklin et al. 2014), is a system-on-chip (SoC) solution which is installed between the network or serial controllers of the PLC. It is capable of capturing network traffic and program updates to ensure that formal safety and liveness specifications are not violated. Being an embedded system, TAIGRA introduces minimal latency, however each module only protects the single PLC in which is it installed.

In the context of monitoring SCADA/ICS safety and security, online runtime verification is the most valuable technique as it allows violations to be caught and alarms triggered in real-time. However, this approach requires great care not to violate fragile timing and functional constraints of the system under observation.

4.3. Runtime Verification of Distributed Systems

The approaches discussed in the previous section were tested on small-scale SCADA systems, typically on a single PLC. For practical runtime SCADA monitoring at industrial scale, the model and verification must cater for many PLCs in a distributed environment. From our review of the literature, little work appears to have been conducted in this area, therefore this section will summarise the current literature on runtime verification of distributed systems in general, with a view to their application to SCADA.

Malakuti et al. (2011), use static code analysis techniques to build a causal model of distributed Java RMI (Remote Method Invocation) calls by wrapping remote calls with a unique identifier. On top of these, a model of the expected behaviour of the application is imposed. In the context of SCADA, their solution

does not address the timing constraints of a safety-critical system and utilises instrumentation in the form of Aspect Oriented Programming (AOP). As stated previously, instrumentation is not a desirable solution in a safety-critical system due to the risk of changing a previously certified system.

Past Time Distributed Temporal Logic (PT-DTL) and DIANA (DIstributed ANAlysis) (Sen et al. 2004) are presented as an efficient way to monitor safety in distributed systems. They focus on a peer-based decentralised approach where each node is aware of other processes' remote states where necessary, but there is no single node with knowledge of the state of the entire system. Therefore their formalism, PT-DTL deals with 'the last known state' of remote processes. Linear Temporal Logic (LTL) is extended into PT-LTL by the inclusion of constructs for "previously", "eventually in the past", "always in the past" and "since". Sen et al. (2004) then extend PT-DTL further by including temporal formulae which refer to the observed states of remote nodes. They identify the following as guidance for efficient distributed runtime monitoring: (1) monitoring should be fast enough to be executed online, (2) local monitors should operate with as little memory overhead as possible and (3) the number of additional messages sent for the purpose of monitoring should be minimal.

DIANA, their implemented monitor would not be directly applicable to an embedded control system as it requires instrumentation of Java bytecode in order to: (1) invoke monitors, (2) track internal variable changes, (3) send messages and (4) respond to messages by updating the local variables representing external variables. This instrumentation would pose challenges to in critical systems as it may violate the requirements for Functionality, Certifiability and Timing as identified in Pike et al. (2011), as previously discussed in **??**. The current solution in Sen et al. (2004) deals only with safety properties not liveness properties. Additionally, the formal safety requirements must be defined upfront manually by an expert.

5. A DISTRIBUTED RUNTIME MONITOR FOR ICS/SCADA

Based upon our review of the literature, we see the following areas of interest to advancing the state of the art in distributed runtime verification in an ICS/SCADA context. In particular we see three intertwined aspects which bring understanding of desired behaviour and runtime information together in the physical setting of ICSs to bring more security to the fore. The three parts are a model (the aspect that attempts to capture the desired behaviour), data sources (the runtime information) and hardware and interfaces (influencing the location of solutions technologies).

5.1. Model

Runtime monitoring examines an actual execution of a system and compares this to some formal model to determine whether the behaviour matches that defined in the model – and needs to alert if there is a mismatch as undesired (or at least unexpected) behaviour has occurred.

For SCADA monitoring, this model must be specified in a formal language which allows for the definition of safety properties in a distributed fashion. A number of formal modelling notations has been proposed (typically some form of temporal or spatial logic). Well known challenges exist in establishing the model. At a basic level an expert could manually define safety constraints. The coverage of this type of model would be limited by the time / knowledge constraints of the expert.

Alternatively, the model could be learned from the behaviours of the system under observation using either artificial intelligence or statistical analysis techniques to learn what 'normal operating' behaviour is. This assumes that the system is currently operating correctly and is limited in that only the states encountered will be learned. It may trigger false alarms or miss safety events during critical error and recovery states which have not previously been learned by the system. Such monitoring is still useful in the detection of subtle semantic attacks leading to a general degradation of equipment such as in the Stuxnet attack. Additionally, they may also be able to detect other kinds of attack such as a DOS attack caused by increased traffic on the network.

A third approach would be to reverse engineer the program logic in each PLC into a formal logic which could be used to generate the monitor. This would allow the monitor to 'lock in' the current logic, assuming that logic is already correct.

A combination of the first and second approach allows for a model which covers the core safety concerns whilst also offering the extra coverage of learning behaviours of which the expert may be unaware. Combining the first and third approach would allow the current program to be extracted and abstracted such that it could be to presented to an engineer to reason about the program and even allow for model checking / theorem solving techniques to validate the safety of the model for runtime verification. A combination of all three approaches would provide the most robust model, it would cater for 'obvious' safety concerns defined by an expert, analyse trends to identify anomalous behaviour and

allow for static model check of the current program to ensure that it meets it's current safety specifications.

However, the anticipated solution needs to go beyond this in that it needs to allow to combine local understanding, generated by a mix of the above methods, with global (i.e. component overarching) understanding to address the complexity of today's SCADA systems. In particular it will need to cope with missing information (not all code could be re-engineered; experts are not all-knowing) and imprecision (some parts of the overall system will be better understood than others) – thus really challenging current logic based models.

5.2. Data Sources

The models allow to understand the system, but for monitoring purposes they depend on availability of real-time data. One possible source of data is analysis of the network traffic itself, by inserting monitors in between the PLCs and the HMI in order to intercept and extract data from the traffic. This is the approach most frequently taken in the literature. Potential issues with this approach are that: (1) it assumes that the network traffic is trustworthy, (2) the monitor can intercept and relay traffic without violating real-time constraints (3) the necessary values appear in network traffic (some values may exist only within the PLC). This approach would also need to cater for different network protocols which may be in use.

A second approach to investigate is the possibility of using a diagnostic port on the PLC to examine the registers directly. A feasibility analysis would be required to ensure that (1) such a diagnostic port features on a sufficient number of PLC models and (2) that the usage of the port does not incur any performance penalty in the operation of the PLC.

A third approach examines values in the two central databases, the first is frequently called the 'current' database which stores the last observed state of registers from the PLC and is used to drive the HMI. The second database is known as the 'historian' and stores historical values for trend analysis and reporting. Since these databases are located closest (and often mirrored in) the corporate network and contain values passed from sensors via the PLC. Therefore whilst they are the easiest to access, they can be considered the least trustworthy option.

The needed solution will access data from different sources, needs to aggregate this in a way that allows to capture a certainty about its correctness and is open to auditing on this account. It further needs to be able to derive conclusions with certainty in the absence of some detailed information as bandwidth limitations or inaccessibility of PLCs might make it impossible to obtain all the data one wants.

5.3. Hardware and Interfaces

A final consideration is the physical hardware of the monitors. Examples from the literature include various embedded solutions built upon FPGA technology, as well as low-cost hobbyist devices such as those in the Raspberry Pi and Arduino families. If our monitors are intended to be deployed alongside the PLC hardware then concerns need to be addresses surrounding (1) cost, (2) powering the devices, (3) networking them and (4) trust, in terms of both reliability and that the devices themselves have not been compromised.

Here we foresee that the ultimate solution will be interfacing with existing components and add hardware to others where possible, but will likely need to make certain decisions remotely while others need to be made locally (quite close to the PLC) and both the models and data source questions raised before need to support such a distributed observation, aggregation and decision making structure.

6. CONCLUSIONS

In this paper, we have reviewed the current literature surrounding the threats, vulnerabilities and existing approaches to securing vulnerable SCADA systems. We then reviewed the literature surrounding current approaches to securing SCADA systems and specifically into usage of a distributed online runtime monitoring approach to detect violations of safety properties. We conclude with suggestions for further research needed to progress the state of the art in the area of online runtime monitoring of a distributed SCADA system.

Having considered the current setting, state-of-the-art and demands and directions of development of ICS/SCADA Systems it seems clear that attacks will be more nuanced and smarter and that only distributed run-time monitoring which provides an overall view of the controlled system (and possibly some of its context) can successfully identify and resolve threats as needed. Much work has been done on individual aspects that can be combined and integrated to move this agenda forward. A further opportunity presents itself in adopting part of the IoT offering (which in turn can gain many benefits from ICS/SCADA work). However, the challenges are manifold and particularly lie in the strong requirement of traceability and accountability in a very distributed system operating in a very hostile environment with very limited resources and spare capacity.

REFERENCES

Anh, P. D. and Chau, T. D. (2009) Component-oriented architecture for SCADA system. In: *Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on*, 422–427.

Association, A. G. et al. (2005) Cryptographic protection of SCADA communications part 1: Background, policies and test plan. Technical Report AGA Report.

Barbosa, R. R. R., Sadre, and R. Pras, A. (2012). A first look into SCADA network traffic. In: *Network Operations and Management Symposium (NOMS), 2012 IEEE*, 518–521.

Beaver, C. et al. (2002) Key management for SCADA. Cryptog. Information Sys. Security Dept., Sandia Nat. Labs, Tech. Rep. SAND2001-3252. Available from http://www.smartgridinformation.info/pdf/4646_doc_1.pdf

Brasser, F. et al. (2015) Tytan: Tiny trust anchor for tiny devices. In: *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 1–6.

Cárdenas, A. A., Amin, S. Sastry, S. (2008) Research challenges for the security of control systems. In: *HotSec*. Available from https://www.usenix.org/legacy/events/hotsec08/tech/full_papers/cardenas/cardenas_html/hotsecHTML.html

Cau, A. (2009). *Interval temporal logic*. Available from http://www.antonio-cau.co.uk/ITL/

Choi, D., Kim, H., Won, D., and Kim, S. (2009). Advanced key-management architecture for secure SCADA communications. *IEEE Transactions on Power Delivery*, 24(3), 1154–1163.

Dawson, R. et al. (2006) SKMA: A key management architecture for SCADA systems. In: *Proceedings of the 2006 Australasian Workshops on Grid Computing and e-Research - Volume 54', ACSW Frontiers '06*, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 183–192. Available from http://dl.acm.org/citation.cfm?id=1151828.1151850

Dionisio Rocha, A., Peres, and R. Barata, J. (2015). An agent based monitoring architecture for plug and produce based manufacturing systems. In: *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*, 1318–1323.

E-ISAC. (2016) Analysis of the cyber attack on the ukrainian power grid. Electricity Information Sharing and Analysis Center, Tech. Rep.

Erdner, T. and Halang, W. A. (2004) A fault tolerant control and sensor network with predictable real time qos. In: *AFRICON, 2004. 7th AFRICON Conference in Africa*, 2, 1229–1234.

Falliere, N., Murchu, L. O., and Chien, E. (2011) W32. stuxnet dossier. White Paper, Symantec Corp., Security Response, 5.

Fovino, I. N., Coletta, A., Carcano, A. Masera, M. (2012). Critical state-based filtering system for securing SCADA network protocols. *IEEE Transactions on Industrial Electronics*, 59(10), 3943–3950.

Franklin, Z. R., Patterson, C. D., Lerner, L. W., and Prado, R. J. (2014). Isolating trust in an industrial control system-on-chip architecture. In: *Resilient Control Systems (ISRCS), 2014 7th International Symposium on*, 1–6.

Giani, A., Sastry, S., Johansson, K., and Sandberg, H. (2009). The viking project: An initiative on resilient control of power networks. In: *Resilient Control Systems, 2009. ISRCS '09. 2nd International Symposium on*, 31–35.

Gu, Z., Wang, C., Zhang, M., and Wu, Z. (2014) WCET-aware partial control-flow checking for resource-constrained real-time embedded systems. *IEEE Transactions on Industrial Electronics*, 61(10), 5652–5661.

Hadžiosmanović, D., Sommer, R., Zambon, E., and Hartel, P. H. (2014), Through the eye of the PLC: Semantic security monitoring for industrial processes. In: *Proceedings of the 30th Annual Computer Security Applications Conference ACSAC '14*, ACM, New York, NY, USA, 126–135. Available from http://doi.acm.org/10.1145/2664243.2664277

Hale, R. W. S. (1988) Programming in temporal logic. PhD thesis, University of Cambridge.

Janicke, H., Nicholson, A., Webber, S., and Cau, A. (2015) Runtime-monitoring for industrial control systems. *Electronics*, 4(4), 995. Available from http://www.mdpi.com/2079-9292/4/4/995

Kalapatapu, R. (2004) SCADA protocols and communication trends. In: *ISA2004*.

Kumar, N. R., Mohanapriya, P., and Kalaiselvi, M. (2014) Development of an attack-resistant and secure SCADA system using WSN, MANET, and Internet. *International Journal of Advanced Computer Research*, 4(2), 627.

Lee, S., Choi, D., Park, C., and Kim, S. (2008) An efficient key management scheme for secure SCADA communication. In: *Proceedings of world academy of science, engineering and technology*, 35, Citeseer.

Leucker, M. Schallhart, C. (2009) A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5), 293–303.

Malakuti, S., Aksit, M., and Bockisch, C. (2011) Runtime verification in distributed computing. *Journal of Convergence*, 2(1), 1–10.

Mao, Y. F., Zhang, Y., Hua, Q., Dai, H. Y., and Wang, X. (2015), A non-intrusive solution to guarantee runtime behavior of open SCADA systems. In: *2015 IEEE International Conference on Web Services (ICWS)*, 739–742.

Mittra, S. (1997) IOLUS: A framework for scalable secure multicasting. *SIGCOMM Comput. Commun. Rev.*, 27(4), 277–288. Available from http://doi.acm.org/10.1145/263109.263179

Morris, T. and Pavurapu, K. (2010) A retrofit network transaction data logger and intrusion detection system for transmission and distribution substations. In: *Power and Energy (PECon), 2010 IEEE International Conference on*, 958–963.

Moszkowski, B. (1984) Executing temporal logic programs. In: *Seminar on concurrency*. Springer, 111–130.

Neukirchner, M., Michaels, T., Axer, P., Quinton, S., and Ernst, R. (2012), Monitoring arbitrary activation patterns in real-time systems. In: *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, 293–302.

Nicholson, A., Webber, S., Dyer, S., Patel, T., and Janicke, H. (2012). SCADA security in the light of cyber-warfare. *Computers & Security*, 31(4), 418–436. Available from http://www.sciencedirect.com/science/article/pii/S0167404812000429

Pal, O., Saiwan, S., Jain, P., Saquib, Z., and Patel, D. (2009) Cryptographic key management for SCADA system: An architectural framework. In: *Advances in Computing, Control, Telecommunication Technologies, 2009. ACT '09. International Conference on*, 169–174.

Pike, L., Niller, S., and Wegmann, N. (2011) Runtime verification for ultra-critical systems. In: *Runtime Verification*. Springer, 310–324.

Piétre-Cambacédès, L. and Sitbon, P. (2008) Cryptographic key management for SCADA systems-issues and perspectives. In: *Information Security and Assurance, 2008. ISA 2008. International Conference on*, 156–161.

Prayati, A., Stathaki, A., Furusjo, E., and King, R. E. (2007) A decision support system with distributed agents for large-scale process control. In: *Control Automation, 2007. MED '07. Mediterranean Conference on*, 1–6.

Sagala, A., Lumbantoruan, D. P., Manurung, E., Situmorang, I., and Gunawan, A. (2015) Secured communication among HMI and controller using RC-4 algorithm and raspberry pi. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 15(3), 526–532.

Sen, K., Vardhan, A., Agha, G., and Rosu, G. (2004) Efficient decentralized monitoring of safety in distributed systems. In: *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, 418–427.

Tsang, P. P. and Smith, S. W. (2008) YASIR: A low-latency, high-integrity security retrofit for legacy SCADA systems. In: *Proceedings of The IFIP TC 11 23rd International Information Security Conference*. Springer, 445–459.

Tupakula, U. and Varadharajan, V. (2014) Techniques for detecting attacks on critical infrastructure. In: *Computing, Networking and Communications (ICNC), 2014 International Conference on*, 48–52.

Vukovic, O., Sou, K. C., Dan, G., and Sandberg, H. (2012) Network-aware mitigation of data integrity attacks on power system state estimation. *IEEE Journal on Selected Areas in Communications*, 30(6), 1108–1118.

Watterson, C. and Heffernan, D. (2008) A runtime verification monitoring approach for embedded industrial controllers. In: *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, 2016–2021.

Wright, A. K., Kinast, J. A., and McCarty, J. (2004) Low-latency cryptographic protection for SCADA communications. In: *Applied Cryptography and Network Security*. Springer, 263–277.