Cyber-Resilient SCADA Systems via Secure State Restoration

Zachary Birnbaum, Matthew Davis, Salman Salman, James Schaffter, Lanier Watkins, Saikiran Yamajala, Shruti Paul

Chapter 1

CYBER RESILIENT SCADA SYSTEMS VIA SECURE STATE RESTORATION

Zachary Birnbaum, Matthew Davis, Salman Salman, James Schaffter, Lanier Watkins, Saikiran Yamajala, Shruti Paul

Abstract Supervisory Control and Data Acquisition (SCADA) systems are widely used by critical infrastructure and are ubiquitous in numerous industries including telecommunications, gas, and manufacturing. SCADA systems are high risk targets for cyber attack given their criticality, interconnectedness, and internet accessibility. Programmable Logic Controllers (PLCs) are one of the primary components of SCADA systems which monitor and provide control instructions to other devices, as well as receive data from sensors. Unfortunately, PLCs today are configured in a persistent manner, i.e. physical devices are configured once and designed to be continuously operational, and are ill-suited to operate in a virtual, dynamic, and cyber resilient environment. Therefore, there is a need for cyber resilient SCADA architectures which can endure and recover from cyber attacks. The presented approach describes a secure way to store system states which are then used by redundant, non-persistent devices during operation and recovery procedures. Successful application of the approach results in a non-persistent, Byzantine fault tolerant, virtual Industrial Control System (ICS) where state and function can be stored and restored securely, resulting in greater system cyber resilience. To demonstrate the efficacy and assess the applicability of the approach, mathematical and timing analyses were conducted. An environment implementing the approach architecture was configured and tested where a cyber attack against non-persistent PLCs resulted in quick identification and subsequent secure restoration with no loss in state or functionality. This approach works as a practical framework for applying cyber resilience with non-persistence and state restoration to SCADA environments.

Keywords: cyber resilience, SCADA cybersecurity, non-persistent systems, ICS virtualization, fault tolerance

1. Introduction

Supervisory Control and Data Acquisition (SCADA) systems are key for operation of modern critical infrastructure and industries such as power generation, transportation, and manufacturing [32]. Unlike many enterprise environments, these complex cyber physical systems have real-time dependencies due to their interactions with the physical world. SCADA systems are usually composed of the following devices:

- Programmable Logic Controllers (PLCs) operate as primary components of a SCADA system, managing operations, driving actuators, and processing sensors.
- Communication Links are used by SCADA systems to distribute information over wide geographical areas. These links commonly use specialized industrial protocols such as Modbus or Common Industrial Protocol (CIP) and often use standard Ethernet and IP protocols.
- Human Machine Interfaces (HMIs) provide operators with a graphical user interface (GUI) to observe and control the system [31].
- Sensors provide operation data to PLCs for processing.
- Actuators move or control a physical device such as a motor, valve, or pump.

Due to their critical and interconnected nature, these systems are increasingly the target of cyber attacks. As demonstrated by Stuxnet and others [22], SCADA systems are not only vulnerable to cyber attack, but a successful attack may cause lasting and irreparable damage. Stuxnet, a self replicating computer virus, was able to infect PLCs used by Iran's nuclear program and destroy over 1000 uranium enrichment centrifuges, significantly impacting Iranian goals. Similar cyber attacks against critical systems have transformed what was once thought to be isolated and secure industrial networks into networks that must be protected and defended in order to assure continued operations.

Securing SCADA systems from cyber attacks is no easy feat. Many classical cybersecurity defensive techniques and technologies such as encryption, firewalls, anomaly detection, and patching are difficult to apply. Something as simple as applying a system update, which happens with regularity in enterprise environments, is challenging due to the high availability demands and critical nature of SCADA systems. Additionally, many SCADA systems were designed and integrated many years ago, and while using

and Technology provides 14 resilience techniques including Diversity, Unpredictability, Non-Persistence, and Redundancy [25].

Cyber resilience techniques and defensive technologies can now be applied to SCADA environments through the use of virtualization. There are numerous companies and integrators which now offer virtual SCADA and Industrial Control System (ICS) components. During a new integration or system update, older SCADA components can be replaced by their virtual counterparts. The virtualization of these critical systems allows for complex cybersecurity techniques to be applied, due to the significantly increasing computing power, available memory, and network bandwidth available. For example, the virtualization of systems allows for them to be restarted on demand in very little time, something that is not commonly found in conventional physical instantiations. Even rebooting the system, a relatively simple cyber defensive action, is a manifestation of cyber resilient non-persistance and is effective against certain types of malware [18].

Unfortunately, applying cyber resilient techniques to SCADA environments is not without its challenges, primarily due to SCADA's real-time nature, high availability requirements, and criticality. For example, when applying a technique such as non-persistance as manifested through a system reboot, minimizing the system downtime is of utmost importance. Additionally, SCADA systems provide an added challenge when applying non-persistent techniques, namely how quickly can positive control be exerted by the system over its physical environment and whether this time delta is within acceptable system limits. This problem of establishing positive control stems from the fact that SCADA systems are a form of control systems, which oftentimes use not only current inputs to control output, but past inputs and outputs as well. The reliance on past states to control current output sits in direct conflict to the cyber resilient technique of non-persistence. Other resilience techniques, such as redundancy or heterogeneity, can be more easily integrated into a cyber resilient solution for SCADA without concerns for past system state.

Novel solutions must be developed to address the problem of applying cyber resilient techniques to SCADA systems. The approach described herein seeks to address a subset of this larger problem. The primary contribution of this paper is a methodology to securely restore state in SCADA environments to enhance system cyber resilience. State storage and restoration is coupled with Byzantine fault tolerant redundant components as building blocks for a cyber resilient SCADA architecture. Furthermore, a representative system is presented demonstrating the efficacy and utility of the approach.

The remainder of this paper is organized as follows: Section 1.2 discusses the background and motivations for the presented work. Section 1.3 provides an analysis of the existing work in the field of SCADA cyber resilience and assured operations. Section 1.4 describes in detail the approach for cyber resilient SCADA systems. Section 1.5 demonstrates the successful implementation on a test system. Section 1.6 discusses the strengths and limitations of the presented method. Lastly, Section 1.7 presents concluding remarks.

2. Background

2.1 Control Theory

The primary objective of any SCADA system is to maintain output status in spite of various disturbance effects [28]. The maintenance of system outputs to achieve a desirable system state is done through feedback or feedforward mechanisms. Feedforward control, typically used by Proportional, Integral, and Derivative (PID) controllers [24], measures disturbances and using these errors will adjust the system control effort. Feedback control measures the difference between the true and desired system states, and using this error will adjust the control effort. Additional information on control systems can be found in [5]. For the remainder of this section, a feedback system is assumed, however similar descriptions can be used for feedforward systems.

Control systems are typically designed in the Laplace domain [28], transforming difficult to solve continuous time domain differential equations into simple frequency domain equivalents. The designed solution is then converted to a discrete time difference equation for modern controller implementation.

For example, a third order continuous time controller can be represented in the Laplace ($\mathcal{L}$) domain as follows:

$$H(s) = \frac{a_2 s^2 + a_1 s + a_0}{b_3 s^3 + b_2 s^2 + b_1 s + b_0} \tag{1}$$

where a_i and b_i are constants, and s is the complex number frequency parameter.

Assuming a constant time step, Δt, and adjusting for causality, the discrete time equivalent represented in the $\mathcal{Z}$ domain is as follows:

$$H(z) = \frac{\alpha_2 z^{-1} + \alpha_1 z^{-2} + \alpha_0 z^{-3}}{1 + \beta_2 z^{-1} + \beta_1 z^{-2} + \beta_0 z^{-3}} \tag{2}$$

where α_i and β_i are constants, and z is the discrete complex time parameter.

The $\mathcal{Z}$ domain controller can be represented as a difference equation as follows:

$$y(k) = -\beta_2 y(k-1) + -\beta_1 y(k-2) + -\beta_0 y(k-2) + \alpha_2 x(k) + \alpha_1 x(k-1) + \alpha_0 x(k-2) \tag{3}$$

where α_i and β_i are constants, y is the output, and k are discrete time steps.

This difference equation can then be easily implemented on a controller, transforming inputs to outputs consistent with the system design. Even in this fairly simple control system example, past

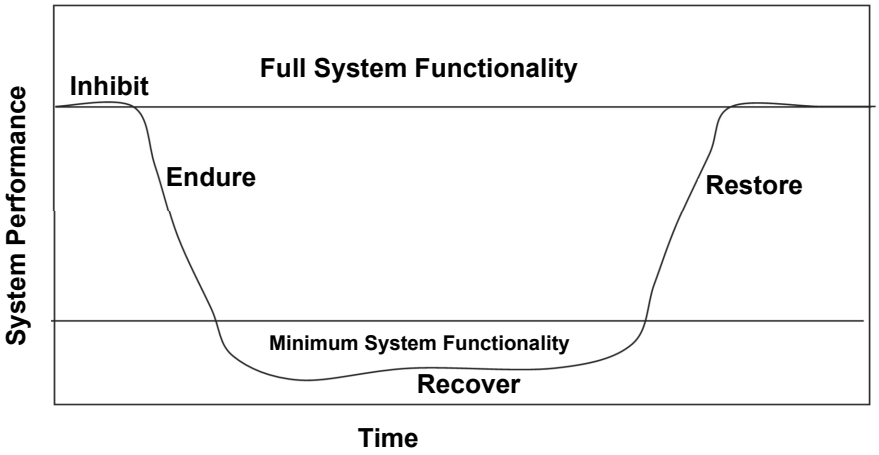

Figure 1. Example Cyber Resilience Capabilities Timeline

are deployed, a single hypervisor will contain many instances of virtual PLCs. Benefits of virtualization include the following:

- Decreased number of physical devices. Co-location of multiple virtual PLCs and HMI processing devices decreases the physical footprint of the SCADA system.
- Simplified development and disaster recovery. Through virtual machine snapshots [26], failures caused by software development errors, cyber attack, or catastrophe can be quickly rolled back to a previously saved copy.
- Security architecture testing. There are many virtual security solutions which are easily deployed to virtual environments for integration into SCADA systems. Not only can security be integrated into the hypervisor, but with virtualization, standard host based solutions can be added directly to the virtual PLCs.
- System updates. Virtual systems are significantly easier to update and patch when compared to their traditional counterparts. A patch can first be tested, and then if functionality is verified, the patch can be integrated into the environment. If any failure occurs due to the update, the system can be rolled back to a previously saved image.

Disadvantages of virtualization include the following:

- Additional system complexity. Adding any new technology to a system increases the difficulty in finding the origin of a system problem or failure.
- Non-standard physical interfaces. Many SCADA environments use proprietary, outdated, or non-standard physical devices to interface with actuators and sensors. There will be challenges interfacing these custom devices with commercial off the shelf (COTS) modern server equipment.
- Increased cyber attack surface. Commercially developed hypervisors add to the attack surface of the system. Successfully attacking the hypervisor may allow an attacker to gain root privileges on all virtual machines contained within the physical server [7]. However, an ongoing area of research is focused on developing cyber secure hypervisors, ensuring that the attack surface is not increased [29].

The approach described herein assumes a virtualized SCADA system. The benefits of virtualization outweigh the disadvantages [27], which can be mitigated by technical maturity and careful adoption. Additionally, there has been an increased effort to research virtual SCADA devices and testbeds [2][3]. This new technology and its application not only increases cost savings and ease of integration, but can also enable cyber resilience. Cyber resilience of SCADA systems is timely and critical given the focus and attention paid to recent cyber attacks. The focus of this paper is to demonstrate the increased cyber resilience of a SCADA system made possible by virtualization.

2.3 Cyber Resilience

Cyber resilience is still a relatively new concept in the cybersecurity field, and can be broken down into several characteristics, which when considered holistically determine the overall cyber resilience of a system [8].

- Inhibit. While technically not an official characteristic of cyber resilience, inhibit capabilities prevent an adversary's ability to cause a cyber effect within the system. These techniques generally fall under the category of traditional cybersecurity but are still important to the overall resilience of the system.
- Endure. Endurance capabilities are the abilities for a system to continue to provide minimum functionality while under cyber attack.
- Recover. Recovery capabilities are the abilities to take any necessary actions in response to degraded system performance.
- Restore. Restoration capabilities are the abilities to regain complete functionality of the system.
- Improve. Improve capabilities are the abilities to adapt enhanced cyber resilience design and implementation to mitigate the anticipated threat.

- Applied Analytics & Monitoring is the continual collection and analysis of system data to identify possible vulnerabilities, adversary activities, and negative impact to the system.
- Heterogeneity is the utilization of a cyber diverse set of technologies in order to minimize the impact of attacks and simultaneously requires adversaries to attack different technologies with different sets of vulnerabilities.
- Distributive Allocation is the positioning of critical assets in order to provide an unpredictable attack surface for the adversary.
- Non-Persistence is the retention of systems for a limited time, resulting in the reduction of an adversary's opportunity to act maliciously and establish a persistent foothold. A persistent system is generally configured once, and then only accessed if troubleshooting or maintenance is required. A non-persistent system is designed to support repeated shutdown, destruction, and initialization.
- Redundancy is the maintenance of multiple protected instantiations of systems, forcing the adversary to achieve cyber effects on multiple targets simultaneously.

Many cyber-resilience techniques, as discussed in Section 1.3, have been explored for enterprise and ICS applicability, implemented in standard and virtual environments. However, the presented approach focuses on SCADA system endurance restoration with a non-persistent, redundant virtual environment.

3. Related Work

There are works related to various aspects of cyber resilience which can generally be classified by the technique, such as non-persistence, distributed allocation, and heterogeneity.

Cardenas et al. [10] provides an overview of the challenges surrounding securing SCADA systems from cyber attack. While the majority of the identified challenges remain unsolved, they identify unique properties of these systems when compared to enterprise environments, namely their legacy nature and real-time reliance.

Melin et al [23] demonstrates that not only can traditional cybersecurity techniques be successfully applied to SCADA systems, but the system itself can be designed to be cyber resilient. This research focuses on designing and applying a framework and testing criteria for system resiliency. A key finding is that by removing the ability to remotely program an individual PLC, overall system resilience would be greatly increased, limiting the attacker's ability to adjust only the system reference which can be easily detected.

Heterogeneity has also been demonstrated as an effective means to provide cyber resilience to both SCADA and enterprise systems. Cox et al. [13] demonstrates that heterogeneity can be an effective tool to provide cyber resilience for systems by using automated diversity to provide high assurance detection to disrupt attacks. The framework executes heterogeneous variants on the same inputs, and monitors their behavior to detect divergence. The successful application of this technique forces an attacker to compromise multiple system variants to achieve their effect. Gearheart et al [17] builds upon the diversity work by creating a set of metrics to measure diversity, demonstrating that text-based features can effectively differentiate software diversity strategies implemented within two open-source diversifying compilers, thereby increasing cybersecurity.

Byzantine fault tolerant systems, a form of distributed allocation or redundancy, have also proved effective in increasing cyber resilience. There has been considerable work in this area, particularly when coupled with virtualization and other resilience enabling technologies. Ahmed et al. [1] demonstrated that Byzantine Fault-Avoidance (BFA) can be implemented in a cloud environment using OpenStack and Software Defined Networking (SDN) by only allowing replicas to live for a short time on a computing platform (hypervisors, hardware, OS) as a form of moving target defense. The primary contributions of their work include a fault avoidance architecture leveraging cloud platform technologies, providing replica refresh algorithms to control platforms' exposure to attack, and a scheme to preserve state while undergoing failure avoidance. Byzantine architectures use a set of n replicas for failure avoidance. Within the replicas, a primary node exchanges consensus messages to a specific replica. The replica node then prepares a reply and commits the response. The specific node that is used can be selected from a pre-prepared Virtual Machine (VM) pool, which is refreshed after a set amount of time or after completing a specific number of transactions. The primary benefit of such a system is that it removes the advantage of time for a possible attacker in that a single VM is not operational indefinitely. The authors then conduct a series of experiments to demonstrate the proposed system. By dynamically using the cloud software stack they were able to minimize the VM's attack exposure window and boot new VMs in under 12 seconds. Unfortunately, while the system has its benefits, a 12 second recovery time is often unacceptable in real-time environments.

A direct application of cyber resilience techniques to SCADA systems was investigated by Babay et al. [6] who created an intrusion tolerant SCADA system which is resilient to system level compromises and network level attacks. Their approach uses the Spines messaging framework, which provides automatic reconfiguration and network flexibility [4]. The authors distribute core SCADA functionality across $3f + 2k + 1$ heterogeneously compiled (multicompiler) replicas, where f is the number of simultaneous acceptable failures and k is the number of proactive recoveries (no failure detected). The replicas are then distributed across multiple cloud servers. The SCADA system was then used in a Pacific Northwest National Laboratory (PNNL) power grid experiment and met all latency and power requirements. The authors do not address how PLC failures are detected or how state can be correctly transitioned to new environments.

Yomamoto et al. demonstrates that using a customized intrusion detection solution and virtualization

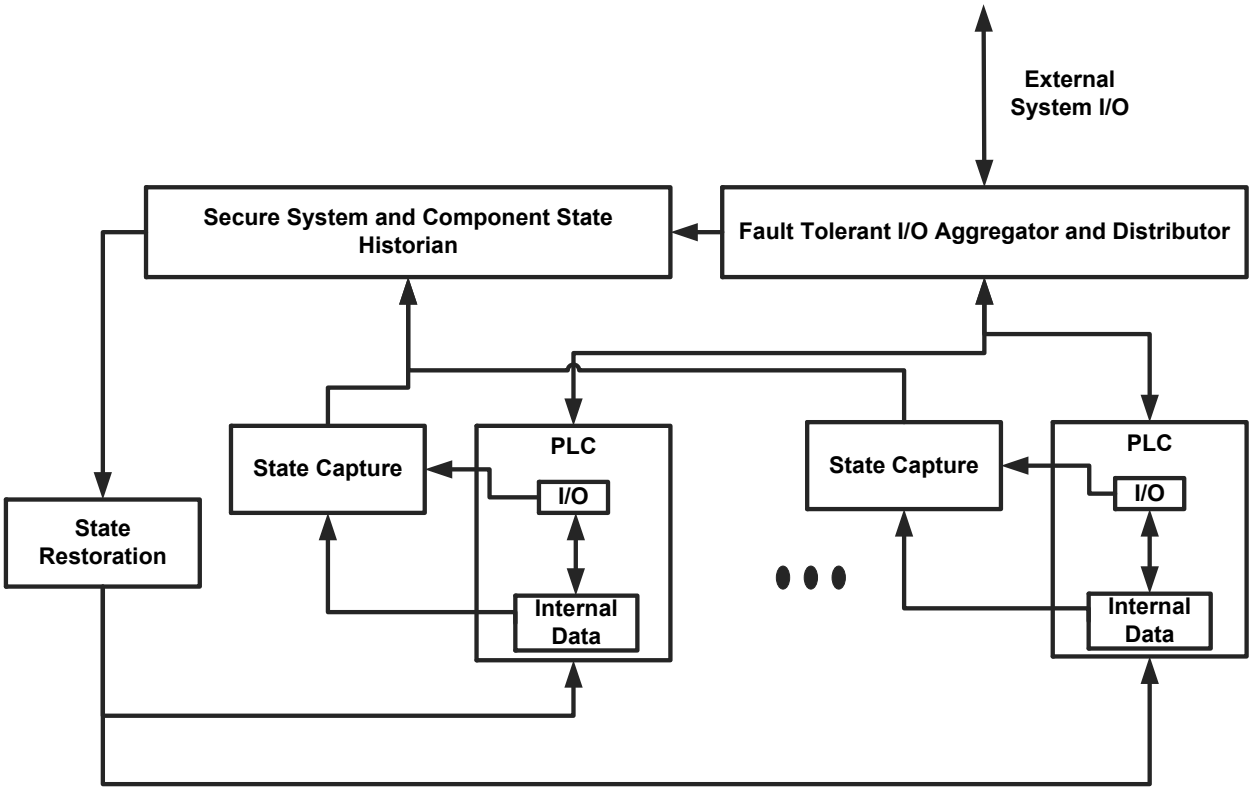

Figure 2. Approach Architecture

4. Approach

4.1 Approach Overview

Rapid and reliable state restoration is an enabling capability of any SCADA system employing a cyber resilient architecture and implementing techniques such as non-persistence. Additionally, employing redundancy further increase system cyber resilience. The combination of state storage and restoration in a non-persistent, redundant environment provides the foundation for the presented approach.

There are several key components of the approach architecture shown in Figure 2 which are as follows:

- State Capture. SCADA component state must be identified and acquired without negatively impacting performance.
- State Storage. After the state has been captured, it must be stored in a secure, immutable, and decentralized fashion.
- State Recovery. When a new, non-persistent SCADA component is added to the network the state must be promptly pushed to the new device to ensure continued operation.
- I/O Aggregator and Distributor. Securely processes the input and output going to and from the redundant, non-persistent SCADA components.
- Non-persistent SCADA components. Redundant PLCs and other devices which can be started, stopped, and restarted with relatively little effort.

These multiple components replace a single traditional SCADA device with a more complex and capable architecture. This new architecture enables continuity of normal system operations due to component redundancy, and ensures rapid state recovery of SCADA components when required. For simplicity of explanation, it is assumed that PLCs are the only SCADA components which will have the added non-persistent and redundant capabilities.

Under normal operation the new architecture functions as follows:

1 Input is sent to the I/O aggregator and distributor via normal communication paths. Typically in SCADA environments this is performed wirelessly, over point to point serial protocols, or via Ethernet.

2 The I/O distributor sends input to n PLCs where it is then internally processed. Ideally these PLCs would each be heterogeneous, further increasing cyber resilience.

3 After a PLC scan and successful computation of the output, the internal PLC data structures and memory are saved to the state storage database (blockchain) [30].

4 The I/O aggregator then processes the output from each PLC using a Byzantine fault tolerant computation to determine consensus [6].

5 The I/O aggregator then stores the input data, consensus, and associated metadata within the blockchain.

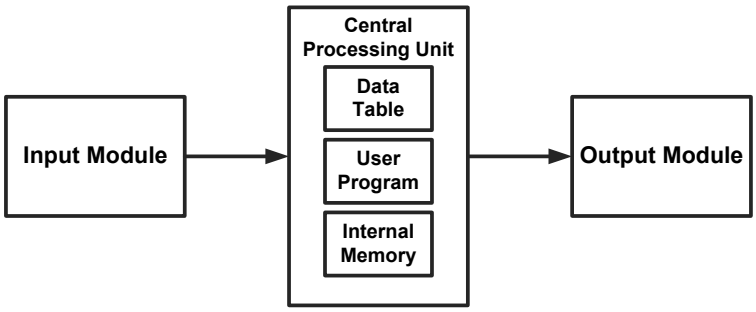

Figure 3. PLC Components

an effect, which would not be detected by the fault tolerant voting scheme, it is recommended that PLCs are destroyed and restored at irregular, operationally small time intervals i.e non-persistence.

The state restoration procedure is as follows for each of the PLCs identified by the conditions above:

1 The PLC is destroyed. In a virtual environment this is a fairly trivial process.

2 A new PLC is initialized to a default state containing no internal data. Ideally the new PLC is heterogeneous from the prior iteration, limiting the likelihood that any prior cyber attack will be subsequently successful.

3 The last known good state of the system is identified using the state database and the corresponding PLC internal memory and data are pushed by the state restoration engine to the new PLC.

The remainder of this section describes in further detail how system state is captured, determined, stored, and then restored in a secure manner.

4.2 Capturing State

In order to capture complete state from a SCADA system, it is important to first identify the necessary locations to collect data. For example, consider a standard PLC system in Figure 3. In order to represent system state it is necessary to collect input data, output data, and any internal memory structures.

With modern SCADA systems utilizing enterprise components for communication [19], the cost to capture input and output information is fairly minimal. If standard Ethernet is present, a physical network tap or modified switch can be used to monitor data going to and from end devices. In the event that the device is virtual, there are equivalent software solutions to capture network data. Regardless of the approach, it is important that detrimental system impacts are not caused by state capture implementation.

The process to capture internal PLC memory and data will vary by vendor [16] and will be highly dependent on the internal structure. For example, if Modbus is the implemented communications protocol and internal data is mapped to holding registers, it is straightforward to request internal data over the network, thereby providing means to capture internal PLC state [12].

Due to their inherent real-time nature, PLCs operate on very predictable scan cycles with each cycle taking several milliseconds. Each scan cycle contains input, program, and output stages [30].

- Input Stage. Input is read by the CPU from the input modules.
- Program Stage. Input and current internal data and memory structures are modified as necessary.
- Output Stage. Any necessary changes in the output modules are made.

It is necessary that any accurate state capture system successfully operates after each scan cycle without degrading overall system performance.

4.3 Determining State

After the I/O and internal state from each device has been captured, the overall system state, calculated using each device, must be determined and any divergent states and devices must be identified. While there are numerous schemes to determine state consensus, Byzantine fault tolerant algorithms are resilient and widely used. The algorithm presented by Castro and Liskov [11] is utilized to provide the fault tolerant voting mechanism where $3f + 1$ replicas must be utilized where f is the maximum number of possible faulty nodes. The agreed upon consensus state by Algorithm 1 is then output, and any malfunctioning devices are flagged for destruction and restoration.

Algorithm 1 Byzantine Fault Tolerance Application

1: Collect output information state from $3f + 1$ devices
2: Determine consensus state as that state common to at least $f + 1$ devices
3: Identify any non-consensus devices for restoration
4: Save consensus to blockchain

state, determining consensus, and adding metadata results in a data block. Decentralized operation prevents an attacker from modifying the shared history from a single location, improving security. There are different Blockchains for the I/O aggregator and distributor, and for each PLC. If PLCs are heterogeneously initialized, then it is required to store a separate Blockchain for each PLC. In the event that all PLCs are homogeneous, then all internal data structures and memory are identical and therefore a single PLC Blockchain can be used.

Each I/O aggregator data block contains the following information:

- Input Value
- Consensus State
- Time
- Iteration number
- Hash value of previous block
- Hash value of all current block data
- List of PLCs agreeing with consensus
- List of PLCs disagreeing with consensus. This list is kept so the state restoration procedure knows which PLCs to restore.

Each PLC data block contains the following information:

- Input Value
- Output Value
- Internal Data and Memory
- Time
- Iteration number
- Hash value of previous block
- Hash value of all current block data

After block information has been computed, the block is appended to a chain containing all preceding blocks. At this point in implementation, state has been securely stored in an immutable fashion and is ready to be restored when required.

4.5 Restoring State

After each data block is appended to the I/O aggregator chain, PLCs which were identified as having disagreed with the consensus are destroyed. Within virtual environments, destruction is often a routine, timely procedure.

A new virtual PLC is then initialized. However, prior to processing new input, the blank internal structures are updated with necessary historical consensus data. It is important to note that the restored state is independent of the specific PLC instance, and represents the agreed upon operating consensus of all participating PLCs. The procedure in Algorithm 2 is conducted to initialize a new PLC.

Algorithm 2 Restore PLC

1: Identify last n consensus states required for successful PLC operation
2: For each identified state, select a random PLC which had an output equal to the consensus state
3: For each identified state, using the selected PLC's Blockchain, pull the internal appropriate information and save within internal data structures to the new PLC

Algorithm 2 described above assumes homogeneous PLCs. Depending on the PLC vendor, virtualization technology, and other factors, writing to internal PLC structures has varying degrees of difficulty. For example, using Modbus, it is possible to write directly to PLC structures using normal communication protocols, whereas other PLCs and devices will undoubtedly require additional ingenuity.

4.6 Approach Summary

The sequence of capturing, storing, and restoring state is the foundation of a cyber resilient system employing non-persistence and redundancy techniques. The following section demonstrates the implementation of the aforementioned approach on a simple architecture.

5. Experimental Verification

Two types of analysis were conducted: (1) mathematical analysis to ensure that the approach increased cyber resilience, and (2) timing analysis to determine the overhead induced by the architecture.

5.1 Mathematical Analysis

Three architectures were analyzed to assess the efficacy of the approach.

- Single standard PLC with no redundancy or state restoration.
- Byzantine fault tolerant architecture without state restoration.
- Byzantine fault tolerant architecture with state restoration.

Each architecture above was analyzed as a process over time to determine when a failure threshold

Table 1. Expected Iterations to Failure for Various Architectures with Threshold Probability $p_t(x) = 0.99$

Architecture	Failure Probability	Redundant PLCs	Expected Iterations to Failure
Single PLC	0.1	0	44
Single PLC	0.2	0	21
Single PLC	0.4	0	10
Byzantine	0.1	1	1243
Byzantine	0.2	1	167
Byzantine	0.4	1	24
Byzantine	0.1	3	504862
Byzantine	0.2	3	5326
Byzantine	0.4	3	82
Byzantine	0.1	5	170381911
Byzantine	0.2	5	141198
Byzantine	0.4	5	239

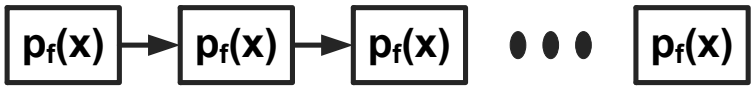

Figure 4. Architecture Iteration Flow

where i is the number of expected iterations until system failure.

For Byzantine architectures, at least $f + 1$ out of $3f + 1$ individual PLC operations must succeed in order for the overall architecture iteration to be successful. Therefore a modified calculation of $p_f(x)$ is required.

$$p(\text{at least } m \text{ out of } n) = \sum_{i=m}^{n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{5}$$

where $m = f + 1$, $n = 3f + 1$, $p = 1 - p_f(x)$, and f is the number of tolerable failures.

The following variables were analyzed while also varying the architectures:

- Probability of architecture failure $p_f(x)$ for a single iteration
- Number of redundant PLCs in a Byzantine fault tolerant architecture

The results are shown in Table 1. As one can see, as the probability of any individual iteration failure increases, the expected system longevity decreases, and Byzantine architectures are more resilient than single PLC systems.

It is important to note that the expected iterations until failure for any Byzantine architectures using state restoration can't be computed. This is because after any successful iteration any individual PLC failures will be corrected and the architecture restored to the correct state prior to any subsequent input. Therefore, each iteration is effectively independent and the overall probability of system failure as inputs are processed will never be greater than $p_f(x)$ regardless of the number of iterations or system uptime. Additionally, the overall system probability will always be less than $p_t(x)$.

However, the increased resilience using state restoration is not without cost. Additional computing resources and time are required to successfully identify and restore any anomalous individual PLCs. The following subsection discusses the impact and time required to both store and restore state.

5.2 Timing Analysis

An analysis of various architectures was conducted to assess the timing impact of the improved system resilience approach. The following elements were instantiated using simple Python programs:

- PLCs
- Fault Tolerant I/O Aggregator and Distributor
- Secure System and Component State Historian

Individual components communicated using the Flask micro web framework [20].

Although the developed environment does not strictly adhere to realistic PLC and SCADA restrictions, it does have the same building blocks. The simulated PLCs process a difference equation much like their industrial counterparts. The remaining virtualized components have no analogous readily available industry equivalent and therefore simulated Python representations were utilized.

A number of individual experiments were conducted to assess the impact of different variables on system timing.

5.2.1 Architecture Complexity.

Depending on the PLC complexity, system redundancy, and if blockchain state storage is utilized, system performance was impacted based on the combination of these factors.

- PLC Complexity. The length of the difference equation for an individual PLC. This is a pseudo approximation

Table 2. Execution Time of Single PLC

Number of PLC Coefficients	Execution Time (Seconds)
1	0.011806
10	0.012572
100	0.013346
1000	0.022429

Table 3. Execution Time of I/O Aggregrator and Distributor

Number of Tolerable PLC Failures	Execution Time (Seconds)
0	0.020581
1	0.040104
2	0.061726
4	0.105565
8	0.186051

Table 4. Execution Time of Secure State Storage (Blockchain)

Experiment Type	Secure Storage (Blockchain) Used	Execution Time (Seconds)
PLC	No	0.009254
PLC	Yes	0.013346
BFT	No	0.020581
BFT	Yes	0.038867

The second experiment assumed a byzantine fault tolerant architecture with a variable number of tolerable failed PLCs, f. All other variables were held constant. The number of coefficients i.e PLC complexity, was set at 100, and no state storage in the blockchain was employed. It can be seen in Table 3 that as the number of nodes required for the Byzantine fault tolerant system increases, so does the time required to process a single input. When comparing a Byzantine fault tolerant system with 0 tolerable failures to a single PLC with equal complexity, there is a 54.2% overhead.

The third experiment examined the impact of state restoration. Both single PLC and Byzantine fault tolerant architectures were assessed with and without state saving after each scan cycle. For this experiment the number of PLC coefficients was held at 100, and the number of tolerable failures was 0. As one can see in Table 4, committing state to the blockchain after each iteration causes significant overhead. For a single PLC the overhead was 44.2% and for a Byzantine system the overhead was 88.8%. The increased overhead for the Byzantine system is due to the need to commit two times to the blockchain, once for the PLC within the architecture, and once after consensus is reached.

5.2.2 Restoration Analysis. Until now, no failures were introduced and only architectural complexity was assessed. In order to assess the timing impacts of restoration, both Byzantine fault tolerant and state capture systems must be enabled. When simulating an attack against a single PLC it is assumed that the attack probability is constant throughout time, and that each PLC iteration is equally likely to be attacked. Unlike Section 1.5.1, the goal is not to determine when the architecture will fail, but to assess timing impacts. Therefore only a maximum of f PLCs can fail where f is the number of tolerable PLC failures. Because at most only f PLCs will fail, the architecture will always return the correct result.

Each architectural experiment has a predetermined probability of failure which is propagated to the PLCs. To determine if a PLC has failed during each iteration, f PLCs generate a random value between 0 and 1. If this value is less than the probability of failure, the PLC will yield an incorrect result, forwarding the wrong answer to the I/O Aggregator. After each iteration, the architecture must then restore those malfunctioning PLCs.

The experiment shown in Table 5 varied the number of tolerable PLC failures while keeping the probability of a single failure constant at 40%. As expected, the greater the number of redundant devices, the greater the expected execution time. This is in part due to the increased number of queries that the I/O aggregator must make in order to achieve consensus, and the increased number of commits to the blockchain.

The experiment shown in Table 6 varied the probability of f PLCs failing while holding all other variables constant. For this experiment the number of tolerable PLC failures was 4 and the number of PLC coefficients was 100. While the expected execution time increases with the probability of failure, the failure rate does not incur as much overhead as other variables. These results indicate that the timing cost to restore a PLC is fairly low. When comparing an architecture with 0% failure probability to one with 100%, restoring all f PLCs can be done with only 28.5% overhead.

5.3 Experiments Summary

The mathematical analysis demonstrated that with the implemented architecture including both redundant PLCs in a Byzantine fault tolerant scheme, and the state storage and restoration capability,

Table 6. Execution Time of I/O Aggregrator and Distributor with Secure Storage (Blockchain) for Varying PLC Failure Rates

PLC Failure Rate	Execution Time (Seconds)
0	0.151104
0.05	0.152491
0.1	0.153756
0.2	0.156992
0.4	0.165552
0.8	0.190867
1	0.194210

each system iteration is now independent. This is important because PLCs use difference equations, so past computations are important for present and future computations. As a result of successful approach implementation, past failures cannot propagate to future operations.

When considering timing impacts of the architecture, the experiments demonstrate that implementing redundancy and blockchain state storage increases overhead. However restoring state to failed PLCs is comparatively fast.

6. Discussion

6.1 Strengths

The primary strength of the approach is a more cyber resilient SCADA architecture than exists today. Once implemented, this approach will mitigate a large variety of common cyber attacks against ICS. Due to the applied resilience techniques, redundancy, and non-persistence, an individual attack against a single PLC will no longer cause irreparable damage to an environment. The approach improves both the endurance and recovery phases of cyber resilience. The developed architecture is simple to implement and can easily replace the control structure within current systems. Additionally, this work applies Blockchain technology and fault tolerant consensus in a novel way, ensuring that system state is computed and stored in a secure manner.

Assessment of this approach suggests that implementation impact is fairly minimal due to the reliance on virtualization and modern computing resources, providing that the computing infrastructure exists within the destination environment. The majority of ICS systems currently deployed run on outdated, physical hardware. As new systems are designed and integrated, the transition to modern virtual environments is likely due to their reduced physical footprint, ease of upgrade, and service collocation compared to physical environments.

6.2 Limitations

While the approach removed reliance on a single PLC for successful system operation, it did add additional cyber attack surfaces. An attacker, rather than targeting the PLC, may now target the consensus algorithm, the blockchain, and the PLC destruction and restoration procedures. It is important that all newly introduced components within the architecture utilize best cybersecurity design practices. There are several improvements which can be made to mitigate the increased cyber attack surface added by these new components. The blockchain can be distributed, and itself made non-persistent, ensuring that in the event a single resilient system is attacked it can be restored from a distributed replica. ICS security and best practices is a rapidly evolving and heavily researched area, therefore it is important that any approach or security solutions are cognizant of the most recent threats and technologies.

Virtualizing PLCs and other SCADA equipment is no easy feat. Not all devices can be virtualized at this time, and additional research is required. This approach was implemented and tested using standard enterprise equipment, i.e. laptops running Python code, and not using an ICS testbed. Further research will test the approach within a representative SCADA environment. Additionally, state restoration within virtual PLCs will vary greatly by vendor. The tested implementation, does not fully emulate an industry grade physical or virtual PLC. Additional research is required to test internal memory and data writing to industry standard PLCs.

While the timing overhead of the system was evaluated for this implementation of the approach, extending this approach to a larger SCADA environment may result in increased latency. As SCADA environments are sensitive to timing, this approach may require adjustment to better scale to larger systems. Along the same lines, if there exists limited computing architecture for these SCADA environments, additional computing equipment may be required to manage a larger virtualized environment. Within certain SCADA architectures, there may be an unnecessarily high number of control computations which, if reduced, provide an opportunity to add additional resilience capabilities. This limitation is exemplified by the utilization of Byzantine fault tolerance, which is known for its overhead in terms of communication bandwidth and number of required replicas. Further study is required to determine if this overhead can be reduced.

Cyber resilience is a new and evolving field. The described approach only focuses on a small subset

7. Conclusions and Future Work

Cyber resilient systems are increasingly important in today's cyber contested environment. There are a multitude of resilience techniques, many of which provide meaningful capabilities in a resilient architecture. Some techniques, such as non-persistence, are challenging to implement in real time environments. Supervisory control and data acquisition (SCADA) systems, industrial control systems (ICS), and other real-time systems require knowledge of past system state to successfully compute current output. These real time systems are omnipresent and it is imperative to ensure they have adequate cyber resilience capabilities. The approach presented herein describes an architecture which securely stores system state by first reaching consensus using fault tolerant methods, then stores state using Blockchain technologies. Individual system components are non-persistent and redundant, leverage virtualization, and upon encountering a possible detrimental cyber event, the affected component can be restored from a baseline and made immediately operational by loading the assured state history.

The approach was tested using an emulated environment and was shown to be effective in preventing permanent system failure due to cyber effects. Future work includes analysis and integration of additional resilience techniques such as heterogeneity. Successful application of resilience techniques has the potential to dynamically change the landscape of SCADA and real-time systems cybersecurity design.

References

[1] N. Ahmed, B. Bhargava, From Byzantine fault-tolerance to fault-avoidance: An architectural transformation to attack and failure resiliency, *IEEE Transactions on Cloud Computing*, 2018.

[2] T. Alves, M. Buratto, F. M. de Souza and T. V. Rodrigues, OpenPLC: An open source alternative to automation, *Proceedings of the IEEE Global Humanitarian Technology Conference*, pp. 585–589, 2014.

[3] T. Alves, R. Das and T. Morris, Virtualization of industrial control system testbeds for cybersecurity, *Proceedings of the Second Annual Industrial Control System Security Workshop*, pp. 10–14, 2016.

[4] Y. Amir, C. Danilov, J. Schultz, D. Obenshain, T. Tantillo and A. Babay, Spines, Distributed Systems and Networks Lab at Johns Hopkins University, Baltimore, Maryland (`www.spines.org/`), 2018.

[5] K. Åström and R. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, Princeton University Press, Princeton, New Jersey, 2010.

[6] A. Babay, J. Schultz, T. Tantillo and Y. Amir, Toward an intrusion-tolerant power grid: Challenges and opportunities, *Proceedings of the IEEE Thirty-Eighth International Conference on Distributed Computing Systems*, pp. 1321–1326.

[7] J. Barrowclough and R. Asif, Securing cloud hypervisors: A survey of the threats, vulnerabilities, and countermeasures, *Security and Communication Networks*, 2018.

[8] D. Bodeau and R. Graubart, Cyber Resiliency Engineering Framework, MTR110237, MITRE Corporation, Bedford, Massachusetts, 2011.

[9] F. Björck, M. Henkel, J. Stirna and J. Zdravkovic, Cyber resilience – Fundamentals for a definition, in *New Contributions in Information Systems and Technologies*, A. Rocha, A. Correia, S. Costanzo and L. Reis (Eds.), Springer, Cham, Switzerland, pp. 311–316, 2015.

[10] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig and S. Sastry, Challenges for securing cyber physical systems, *Workshop on Future Directions in Cyber-Physical Systems Security*, 2009.

[11] M. Castro and B. Liskov, Practical Byzantine fault tolerance, *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, 1999.

[12] Control Solutions Minnesota, Modbus 101 - Introduction to Modbus, St. Paul, Minnesota (`www.csimn.com/CSI_pages/Modbus101.html`).

[13] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong and J. Hiser, N-Variant systems: A secretless framework for security through diversity, *Proceedings of the Fifteenth USENIX Security Symposium*, pp. 105–120, 2006.

[14] M. Crosby, Blockchain technology: Beyond Bitcoin, *Applied Innovation Review*, pp. 6–10, 2016.

[15] G. Engel and M. Mumcouglu, Method for Detecting Anomaly Action within a Computer Network, U.S. Patent No. 2014/0165207 A1, June 12, 2014.

[16] J. Gaspar, Schneider P57 PLC Memory Analyze Log Data in the PLC + Analysis with Matlab, Universidade de Lisboa, Lisboa, Portugal (`users.isr.ist.utl.pt/~jag/course_utils/plc_log/plc_data_log.html`), 2019.

[17] A. Gearhart, P. Hamilton and J. Coffman, An analysis of automated software diversity using unstructured text analytics, *Proceedings of the Forty-Eighth Annual IEEE/IFIP International*

[21] E. Johansson, Virtualization in Control Systems Possibilities and Challenges, SANS Cyber Security Summit, 2009.

[22] D. Kushner, The real story of Stuxnet, *IEEE Spectrum*, Vol. 50(3), pp. 48–53, 2013.

[23] A. Melin, E. Ferragut, J. Laska, D. Fugate and R. Kisner, A mathematical framework for the analysis of cyber-resilient control systems, *Proceedings of the Sixth International Symposium on Resilient Control Systems*, pp. 13–18, 2013.

[24] P. Nachtwey, Feed forwards augment PID control, *Control Engineering*, Downers Grove, Illinois (`www.controleng.com/articles/feed-forwards-augment-pid-control/`), March 31, 2015.

[25] R. Ross, M. McEvilley and J. Oren, Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems, NIST Special Publication 800-160 (withdrawn), National Institute of Standards and Technology, Gaithersburg, Maryland, 2016.

[26] M. Rouse, VMware Snapshot, TechTarget, Newton, Massachusetts (`searchvmware.techtarget.com/definition/VMware-snapshot`), 2012.

[27] J. Sahoo, S. Mohapatra and R. Lath, Virtualization: A survey on concepts, taxonomy and associated security issues, *Proceedings of the Second International Conference on Computer and Network Technology*, pp. 222–226, 2010.

[28] V. Skormin, *Introduction to Automatic Control*, Linus Publications, Ronkonkoma, New York, 2009.

[29] J. Szefer, E. Keller, R. Lee and J. Rexford, Eliminating the hypervisor attack surface for a more secure cloud, *Proceedings of the Eighteenth ACM Conference on Computer and Communications Security*, pp. 401–412, 2011.

[30] Technology Transfer Services, The Basics of PLC Operation, Tampa Florida (`www.techtransfer.com/blog/basics-plc-operation/`), September 9, 2014.

[31] Trend Micro, Industrial Control System (`www.trendmicro.com/vinfo/us/security/definition/industrial-control-system`).

[32] G. Williamson, OT, ICS, SCADA - What's the difference?, *KuppingerCole Analysts*, July 7, 2015.

[33] S. Yamamoto, T. Hamaguchi, S. Jing, I. Koshijima and Y. Hashimoto, A hot-backup system for backup and restore of ICS to recover from cyber-attacks, in *Advances in Human Factors, Software, and Systems Engineering*, B. Amaba (Ed.), Springer, Cham, Switzerland, pp. 45–53, 2016.