Deploying Intrusion-Tolerant SCADA for the Power Grid

Amy Babay, John Schultz, Thomas Tantillo, Samuel Beckley,
Eamon Jordan, Kevin Ruddell, Kevin Jordan, and Yair Amir

Johns Hopkins University — {babay, tantillo, sbeckle2, yairamir}@cs.jhu.edu
Spread Concepts LLC — {babay, jschultz, yairamir}@spreadconcepts.com
Resurgo LLC — {eamon.jordan, kevin.ruddell, kevin.b.jordan}@resurgo.net

Technical Report CNDS-2019-1 - January 2019
http://www.dsn.jhu.edu

***Abstract*—While there has been considerable research on making power grid Supervisory Control and Data Acquisition (SCADA) systems resilient to attacks, the problem of transitioning these technologies into deployed SCADA systems remains largely unaddressed. We describe our experience and lessons learned in deploying an intrusion-tolerant SCADA system in two realistic environments: a red-team experiment in 2017 and a test deployment in a power plant in 2018. These experiences resulted in technical lessons related to developing an intrusion-tolerant system with a real deployable application, preparing a system for deployment in a hostile environment, and supporting protocol assumptions in that hostile environment. We also discuss some meta-lessons regarding the cultural and interpersonal aspects of transitioning academic research into practice in the power industry.**

I. Introduction

Because of their critical importance to modern society, power grid systems present a high-value target for attackers, and recent events have shown that these systems are in fact being targeted by dedicated nation-state-level attackers. This serious emerging threat has led to considerable research on protecting power grids (e.g. [1]–[3]), and particularly on making the Supervisory Control and Data Acquisition (SCADA) systems that provide their core monitoring and control capabilities resilient to attacks. While the body of academic research on this topic has produced innovative solutions, inventing SCADA systems capable of withstanding sophisticated attacks and operating correctly even while partially compromised (e.g. [4]–[8]), the problem of transitioning these technologies into deployed SCADA systems remains largely unaddressed.

In this paper, we describe our experience and lessons learned throughout four years of working to bridge the gap between academic research and realizing the goal of an intrusion-tolerant power grid. This process started when we were convinced to apply our work on resilient clouds to the power domain: after all, if there is no power, even the best cloud protocols are irrelevant. Based on our work on intrusion-tolerant cloud networking and consistent state, we developed Spire, an intrusion-tolerant SCADA system for the power grid [7]. Spire replicates the SCADA master using Byzantine fault-tolerant replication with performance guarantees under attack, employs diversity and proactive recovery to provide protection over a long system lifetime, and uses an intrusion-tolerant network infrastructure to resiliently connect the system components. This system is complemented by MANA, a machine-learning-based network traffic analyzer and intrusion detection system (IDS) that we developed to detect anomalies and provide the situational awareness essential to an effective defense.

While the journey toward an intrusion-tolerant power grid is still ongoing, in this paper we describe the experience of deploying our intrusion-tolerant SCADA system and IDS in two realistic environments: a red-team experiment in 2017 and a power plant test deployment in 2018. During the week-long red-team experiment, the system successfully withstood attacks from a nation-state-level hacker team. During the test deployment, the system managed a small power topology in a "mothballed" steam-turbine power plant that had active control systems connected to the grid but was not generating power at the time. To the best of our knowledge, this is the first time an intrusion-tolerant SCADA system was tested by a nation-state-level red team, and the first time such a system was deployed (in a test) in an actual power installation.

Overall, our development and deployment experience resulted in two types of lessons. First, it resulted in technical lessons related to developing an intrusion-tolerant system with a real deployable application (a SCADA system, as opposed to a generic replication protocol), preparing a system for deployment in a hostile environment, and supporting protocol assumptions in that hostile environment. Second, it resulted in meta-lessons related to the cultural and interpersonal aspects of transitioning academic research into practice in the power industry, which in our experience is considerably harder than in the cloud and Internet domains.

At the technical level, the lessons we learned in developing and preparing the system centered on the relationship between the application state and the replication protocol, as well as the need for a holistic end-to-end view of intrusion tolerance to resist attacks at all levels of the system. The technical

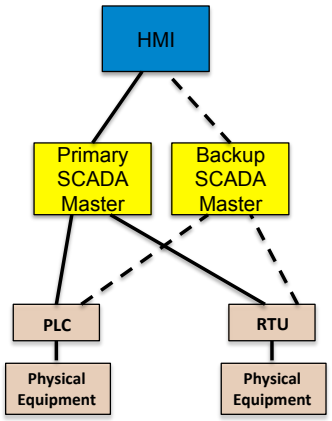

Fig. 1. Conventional SCADA system architecture.

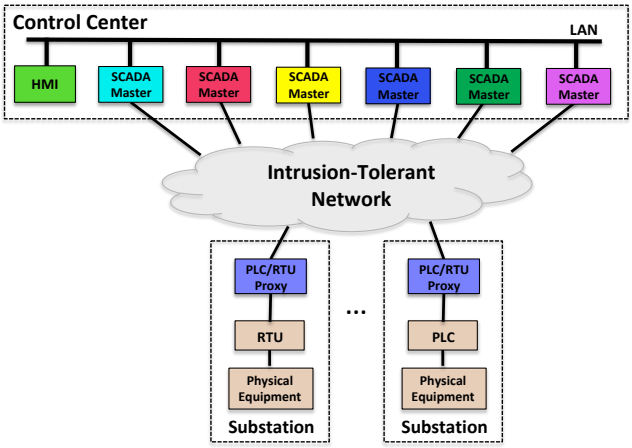

Fig. 2. Spire system architecture using six replicas (to withstand one intrusion and support one proactive recovery).

lessons learned in subjecting the system to a red team attack underscored the need for resilience at all levels of the system, exposing the network and operating system levels as the first targets for attack, and suggested techniques for code compilation and deployment that can increase the work the attacker must do to learn about the system.

Our experience also highlighted the importance of situational awareness for the SCADA system operator. While intrusion tolerance can effectively mask failures and attacks, it is important to present the system's knowledge about ongoing attacks and anomalies directly to the operator to allow them to react and take action. We have found machine learning to be an effective approach for intrusion detection in SCADA environments, due to its ability to passively identify custom and zero-day attacks that do not have known signatures.

At the cultural, meta-lesson level, we gained considerable insight into how to execute a successful short-term experiment in a real environment as well as what is required for the ultimate transition of research into practice in the power industry. These lessons highlighted the need to earn the trust of power company engineers and decision makers as a precondition for any installation on their premises, the need to learn the vocabulary of an entirely different industry to effectively translate ideas, and the need for an incremental deployment plan to make progress in a highly conservative ecosystem.

The remainder of the paper is organized as follows: Section II provides an overview of our intrusion-tolerant SCADA system to give context on exactly what was deployed. Section III discusses design choices that were made to prepare the system for attacks and deployment. Section IV describes the red team experiment conducted in March and April 2017. Section V describes the power plant test deployment conducted in January and February 2018. Section VI summarizes the key lessons from both deployment experiences. Section VII discusses related work, and Section VIII concludes the paper.

II. System Overview

Conventional SCADA system architectures include a SCADA master, Human Machine Interface (HMI), and several Programmable Logic Controllers (PLCs) and/or Remote Terminal Units (RTUs) that interface with the power equipment. Figure 1 shows the architecture of such a system. The SCADA master is the central control server, responsible for collecting status updates from the RTUs and PLCs. The SCADA master uses that information to determine the state of the system, make control decisions, and display the status to the human operator through the HMI. The system operator can use the HMI to issue manual supervisory commands, which the SCADA master processes and sends to the RTUs and PLCs. Due to the SCADA master's importance, a primary-backup architecture is often used to ensure that if the primary SCADA master fails, the backup can take over and restore operation.

Spire, the intrusion-tolerant SCADA system that we have developed, is designed to overcome system-level attacks that can allow an adversary to compromise a SCADA master as well as network-level attacks that aim to disrupt communication between the components of the SCADA system [7], [9]. Its architecture is shown in Figure 2.

At the system-level, it overcomes compromises of the SCADA masters using Byzantine Fault Tolerant replication, where $3f + 1$ total replicas can be used to maintain correct operation in the presence of f compromised replicas [10]. Spire specifically uses the Prime replication engine to provide both safety and latency guarantees under attack [11].

However, if all replicas are identical, intrusion-tolerant replication is not effective: an attacker who compromises one replica can reuse that same exploit to compromise all of the replicas. Therefore, we use the MultiCompiler [12], [13] to diversify the replicas' attack surface by introducing random changes at compile time. These changes do not affect the overall behavior of the program, but change its layout in a way that makes it extremely unlikely that the same exploit will succeed in compromising any two distinct variants.

While diversifying replicas forces an attacker to develop a different exploit for each replica, a dedicated attacker with sufficient time and resources can eventually craft enough distinct exploits to breach the system guarantees. Therefore, we use proactive recovery [10], [14], [15] to periodically take each replica down and restore it to a known clean state with a new diverse variant of the code. Because a replica undergoing proactive recovery is temporarily unavailable, supporting proactive recovery requires more total replicas: to withstand f intrusions when k replicas may be simultaneously undergoing proactive recovery, a total of $3f + 2k + 1$ replicas are needed [15] (6 replicas for $f = 1, k = 1$, as in Figure 2).

At the network-level, Spire uses the Spines intrusion-tolerant network to provide authenticated, encrypted, and resilient communication between the system components [16]. To connect existing PLCs and RTUs to the network, we use a proxy that limits their network attack surface. Their typical, insecure industrial communication protocols, such as Modbus or DNP3, are used only on the direct connection between the PLC or RTU and its proxy, which, ideally, can simply be a wire. The proxy communicates with the rest of the system over the secure and intrusion-tolerant Spines network. HMI communications are similarly protected by a secure proxy.

The system also includes an intrusion detection and situational awareness component called Machine-learning Assisted Network Analyzer (MANA). MANA translates network packet capture into data inputs for machine learning evaluation and alerts users in near real-time of any highly correlated anomalous or malicious activity. Network activity is monitored from a situational awareness board tailored for power plant engineers and can be viewed as part of the HMI.

III. Design Decisions for Deployment

Designing our system to withstand red team attacks led us to take a broader view of intrusion tolerance that considers not just the replication protocol but also the SCADA master that runs on top of it, lower level network and operating system protection, and intrusion detection and situational awareness.

A. SCADA Master Application

Two key differences separate our work on creating a deployable replicated SCADA system from standard BFT replication: it supports a real application that is more complex than the basic databases normally used to evaluate BFT protocols, and the application reflects physical state in the real world. Therefore, the challenge is to ensure that the replicas are not only consistent with one another, but also that their state correctly reflects the real world.

This has two consequences for the system design. First, it requires signaling between the replication protocol and the SCADA application. The replication protocol orders updates consistently, but the SCADA master applies the updates and maintains the application-level state. Therefore, after partitions or proactive recoveries, it is not sufficient to perform catchup and state transfer at the replication-protocol level. The replication layer signals the SCADA master that an application-level state transfer is required, and the SCADA masters must then execute a state transfer protocol at the application level.

Second, due to the cyber-physical nature of the system, the current state of the RTUs and PLCs represents the ground-truth system state, and SCADA masters can recover this state by polling the field devices. This interesting feature opens up the possibility of recovering from temporary assumption breaches in a way that is not possible for generic BFT replication. If enough replicas crash and lose their state such that it is no longer possible to recover the system state from the remaining correct replicas, the system can automatically reset and rebuild the state by contacting the field devices. In contrast, a traditional BFT system cannot recover from this situation. Note that this property only applies to the SCADA master's view of the active system state; SCADA historians are more similar to traditional database applications and cannot recover historical state automatically after an assumption breach.

B. Low-level Protection

While BFT replication overcomes SCADA master compromises, it relies on several assumptions that must be supported to provide the promised resilience; if an attacker can subvert these assumptions, the intrusion-tolerant protocols are not useful. First, the assumption that no more than the tolerated threshold of replicas (f) will be compromised must be supported by securing the operating system. While we employ application-level diversity, if all replicas run an operating system with known vulnerabilities, those can be exploited by an attacker to gain control of the entire system. Because of this, we deploy all system components (SCADA master replicas, HMI, RTU/PLC proxy) on the latest minimal CentOS server installations. This required considerable work to port system components designed to run on Ubuntu desktop installations, namely the HMI graphics packages and the PLC communication libraries. The CentOS server is essentially closed by default, with only external communication that is specifically allowed being permitted, while the Ubuntu desktop runs many preinstalled services and has an open philosophy by default.

Second, there is an assumption that system components can communicate. Specifically, the replication protocol assumes that $2f + k + 1$ correct replicas can communicate with one another, and the SCADA system assumes that at least one correct SCADA master can communicate with the HMI and RTU/PLC proxies and that the proxies can communicate with their RTUs and PLCs. We support this assumption through a secure network setup. As the first step, we configured the firewall of each machine to block all incoming and outgoing traffic other than the specific IP address and port combinations used by our protocols and turned off IPv6 (since we were not using it). Next, we took steps to prevent man-in-the-middle attacks: on each machine, we set up a static mapping of MAC addresses to IP addresses and turned off the default ability for a NIC to answer ARP requests for an IP address assigned to another NIC on the same machine. On the switch, we configured a static mapping of MAC addresses to switch ports.

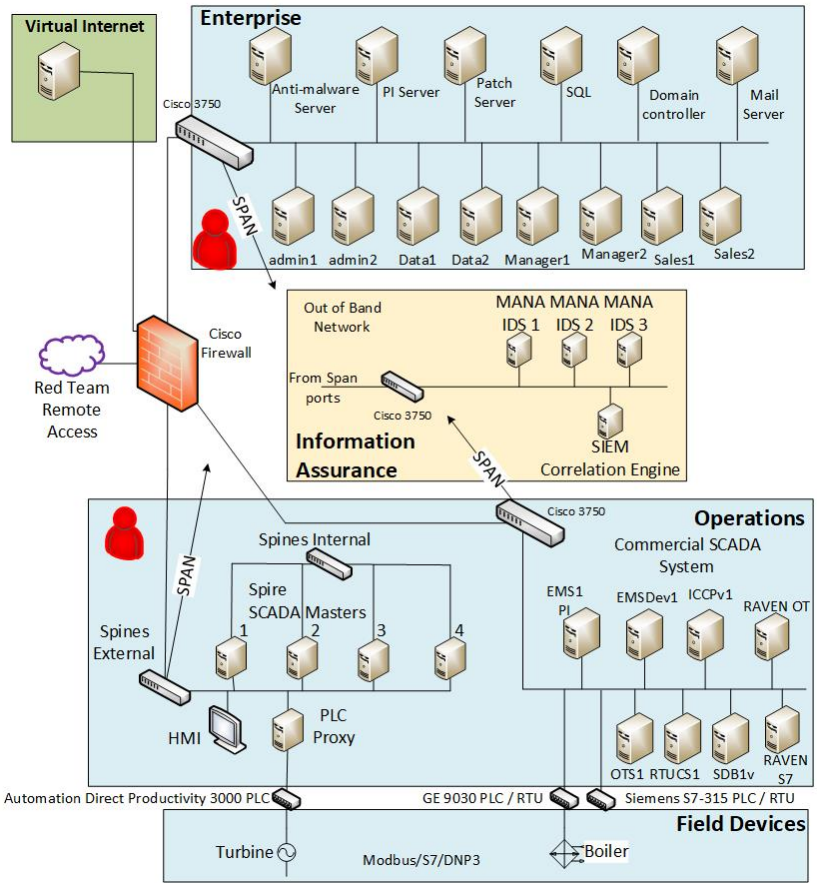

Fig. 3. Experimental setup for red team exercise.

To provide further defense-in-depth, we physically isolate the network used for the SCADA masters' replication protocol from the external network used to communicate with the other system endpoints. This prevents an external attacker from disrupting the replication protocol; they must first compromise one of the replicas to gain network access. For the connection between the PLC proxy and its PLC, we use a physical cable, as opposed to a logical connection through a network switch to ensure that it is not subject to any outside interference.

In Section IV, we show how these design choices were crucial in preventing an experienced red team from interfering with our SCADA system's operation.

C. Operations-Based Intrusion Detection

Due to the operational nature of SCADA environments, network intrusion detection must be applied differently than in traditional enterprise networks. First, the monitoring systems must be completely non-invasive so that the availability of SCADA systems is never in doubt; power plant engineers approved the use of the MANA IDS only because it operated out-of-band, receiving a passive network traffic packet capture. Second, the IDS must handle the myriad of custom and proprietary protocols used by the equipment vendors. Traditional signature-based and deep-packet traffic inspectors are not equipped to effectively inspect unknown, proprietary protocols. Also, as SCADA systems start to add encryption (as our intrusion-tolerant protocols do), traditional network IDS will have even less utility. To overcome this obstacle, MANA uses machine learning and anomaly-based intrusion detection methods, which do not rely on proprietary protocol knowledge or unencrypted traffic. Our preparation for each deployment involved training the machine learning models on the relevant networks.

IV. Red-Team Experiment

In April 2017, our system went through a red-team experiment at Pacific Northwest National Laboratory (PNNL), where an experienced hacker team from Sandia National Laboratories with nation-state-level expertise attacked both a commercial SCADA system set up according to NIST-recommended best practices and our intrusion-tolerant SCADA system.

A. Setup and Preparation

Figure 3 shows the network architecture that was set up by PNNL to model a typical power company network with input from the power company hosting the test deployment in Section V. This architecture includes an *enterprise* network that hosts the SCADA historian (PI Server) as well as other machines used in day-to-day business operation. The enterprise network is separated by a firewall from the

operations network where the SCADA system operates and communicates with field devices.

This experiment used two parallel operations networks: the first (on the right in Figure 3) hosted a commercial SCADA system, while the second (on the left in Figure 3) hosted Spire. Spire was configured with four SCADA master replicas to withstand one intrusion (note that the system was not set up to support automatic proactive recovery, as six replicas are needed to support proactive recovery with bounded delay). As described in Section III-B, replicas communicated with one another on the isolated *Spines Internal* network and with the other system components on the *Spines External* network. A PLC using the Modbus communication protocol was connected to the network through a direct cable connection to the *PLC proxy*.

The MANA IDS was run separately on an out-of-band network that received the network packet capture from the enterprise and two parallel operations networks. Due to the distinct network characteristics of the three networks, we chose to run three independent MANA instances, labeled MANA 1-3 in Figure 3, and to develop three specific network models instead of a single generic one.

Prior to the deployment, we received a specification of the PLC from the PNNL engineers. To integrate the PLC with our system, we developed a new HMI shown in Figure 4. This HMI displayed an (emulated) power topology controlled by the PLC, which consisted of seven breakers managing the flow of power to four buildings. We also integrated the scenario into our SCADA master so that it could maintain and transfer the state correctly. In addition to this physical PLC, Spire controlled ten emulated PLCs modeling power distribution to several substations and remote sites.

The deployment began with one week of on-site setup and integration. During this time, we installed our machines on the operations network as shown in Figure 3, performed the low-level network security steps described in Section III-B, and trained the MANA IDS on the baseline traffic of both the commercial SCADA system and our intrusion-tolerant SCADA system. The machine learning IDS model training was based on a 24-hour network packet capture that occurred toward the end of the setup week, once the three networks had been setup and finalized. Ideally, network traffic collection should occur for a longer period to ensure all traffic characteristics are accounted for, but the experiment timeline only allowed for one day. On-site, we were also required to develop an automatic update generation tool for Spire that would cycle through the breakers, flipping each periodically in a predetermined cycle that the red team would attempt to disrupt.

The rules of engagement for the experiment were that once the systems were set up and the experiment began, we were able to passively monitor the system's activity but were not allowed to take any action.

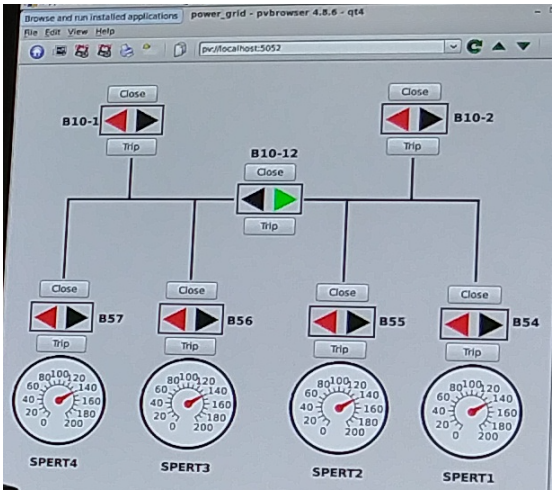

Fig. 4. HMI visualization of power topology for red team experiment.

B. Red Team Experience

The Sandia red team was first given access to the enterprise network and began their attacks on the commercial SCADA system. While the red-team was not expected to be able to cause any damage from that position (due to the firewall separating it from the operations network), surprisingly, within only a few hours, they were able to access the operations network and perform a memory dump of the PLC to obtain its configuration. They then uploaded modified configuration files, enabling them to control the PLC.

In the next stage, the red team was given direct access to the operations network of the commercial system. From there, they were additionally able to disrupt communication between the HMI and the SCADA server, sending modified updates to the HMI and preventing correct updates from being received. These successful attacks clearly demonstrated that the nation's power grid is vulnerable; current best practices provide only weak protection against a nation-state-level attacker.

The red team next attempted to attack the Spire system, starting from the same position in the enterprise network from which they had taken control of the commercial system's PLC. However, after a couple hours, they reported that they had no visibility into the system and asked to be placed directly on the operations network. Over two days, we observed the red team launching network attacks including port scanning, ARP poisoning, IP address spoofing, and denial of service attempts involving bursts of traffic. Other attacks that were not directly observable may have been attempted as well. However, due largely to the secure network setup described in Section III-B and Spines authentication and encryption of all traffic, none of these attacks were successful. The static mappings of MAC addresses to IP addresses and switch ports were especially important in preventing the man-in-the-middle attacks that the red team used in the commercial system, as was the architectural choice of placing the PLC behind a secure proxy, rather than directly on the network.

Finally, on the third day of the experiment, we conducted an

excursion in which the red team was given gradually increasing control of one of the SCADA master replicas (a situation Spire is designed to withstand) as well as access to Spire's source code. This excursion began with user-level access to one replica. The red team first stopped the Spines daemons running on this replica, preventing it from communicating with the rest of the system, but this had no effect; the replicated system can tolerate the loss of any one replica. When the red team restarted the Spines daemon, they ran a custom version they had modified to exploit a vulnerability found in the latest (at the time) open-source version of the Spines codebase; however this had no effect due to newly added encryption that prevented the modified daemon from communicating with the other replicas.

The red team then tried to gain root-level access through known exploits of a shared memory vulnerability in the Linux kernel (dirtycow) and the SSH daemon, but neither was successful due to the use of the latest version of CentOS that had removed those vulnerabilities.

In their final user-level attack, the red team patched the Spines binary on their replica to add in the exploit they had discovered. While this patched version was accepted as a valid member of the network, the attack did not have an effect on the other Spines daemons, as it was in a portion of the code that is disabled when Spines is run in intrusion-tolerant mode.

At that point, the red team was given root access to the machine as well as the latest system source code. They primarily focused on Spines, testing the code in their own lab and attempting attacks there to try to break the fairness properties of the intrusion-tolerant network as a trusted member of the network. Despite this level of access, the red team was still unable to disrupt Spire's operation, demonstrating the effectiveness of its intrusion-tolerant approach. While this result does not mean that given more time, the red team would not have been able to cause damage, it shows that there is a significant difference between current industry best practices and a research-based solution designed to withstand sophisticated system and network attacks.

V. Power Plant Test Deployment

In January 2018, we deployed our system in a "mothballed" steam-turbine power plant. While this plant was not actively generating power (as it was no longer needed to meet demand) it had active control systems that were connected to the grid. The goal of this test deployment was to verify that the system could function in a power grid environment without degrading SCADA performance and without adverse effects on the other power plant systems.

The deployment again began with a week-long setup and integration period. While the red team experiment described above involved controlling a real PLC with emulated breakers, in this deployment the Spire intrusion-tolerant SCADA system controlled a real PLC connected to real breakers. The engineers at the power plant created a subset of the topology shown in Figure 4, including the three breakers on the left of the figure (B10-1, B57, B56). During the setup period, we again built a new HMI to display this power topology and integrated its state into our SCADA master.

This deployment included six diverse SCADA master replicas to simultaneously support one intrusion and one proactive recovery while maintaining continuous correct operation with guaranteed performance (bounded delay for update execution). A PLC proxy was used to communicate with the PLC using the Modbus communication protocol, similarly to the setup of the red team experiment.

In addition to the physical part of the system, there was again an emulated portion, including the same ten emulated PLCs as in the red team experiment (for the larger distribution scenario), as well as six new emulated PLCs modeling a power generation scenario that we created and adapted based on input from the plant engineers.

MANA was set up similarly to the red team experiment but only received network traffic from the operations network (and not from an enterprise network). The plant networked systems employed proprietary SCADA communication protocols that utilized short constant system updates that proved to be ideal for machine learning and anomaly modeling; training required only a single 12-hour packet capture.

After setup and integration with the power plant, Spire and MANA were continuously deployed without interruption or adverse effects on the plant systems for six days. Spire successfully managed the small power topology (along with the two emulated scenarios), displaying the system status and allowing it to be controlled via HMIs in three locations throughout the plant. MANA provided power plant engineers with complete network visibility and situational awareness.

On the last day, plant engineers deployed a measurement device to evaluate the end-to-end reaction time of the commercial SCADA system in the power plant and of Spire. The device periodically flipped a breaker and used two sensors to detect when the HMI screens of the two systems updated to reflect the change. For this test, we adapted the HMI to include a large box that changed from black to white based on the breaker state so that the sensor could easily detect the HMI update. The measurements showed that Spire successfully met the timing requirements of the plant engineers, and was even able to reflect changes more quickly than the commercial system.

VI. Lessons Learned

Our experiences deploying intrusion-tolerant SCADA resulted in lessons in three dimensions: technical lessons, lessons related to conducting a successful red team experiment and deployment, and longer-term lessons related to transitioning intrusion-tolerant technology in the power industry.

A. Technical lessons

The low level setup in Section III-B is far from the topics considered in most academic papers, but all of these steps need to be taken before sophisticated intrusion-tolerant protocols can even have a chance to be relevant. If an attacker can circumvent the protocols at a lower level, they can break the (often implicit) assumptions the protocols rely on.

Our red team experience highlighted the point that the network is a major attack vector that is often overlooked. Low-level network attacks do not require any protocol- or domain-specific knowledge and are a common strength of experienced hacker teams. We saw that the red team's first line of attack was to attempt low-level man-in-the-middle and denial of service attacks; if we had not performed the low-level network setup described in Section III-B (including both the switch set up and the architecture of using two separate networks and putting the PLC behind a proxy), the red team would likely have been able to succeed in at least causing a denial of service without even attempting attacks at the Spines or SCADA system levels.

Similarly, at the operating system level, our effort in porting all system components to the latest minimal CentOS server version was important in preventing the red team from easily escalating their privilege level once they were given access to one of the replicas.

From a defense-in-depth perspective, we also learned that we could have improved the way we compiled and ran our code to make the attackers' job harder. One factor that helped the red team patch the Spines binary with their attempted exploit was the fact that we had compiled the code to include debugging symbols. While stripping symbols from the executable would not prevent the red team from patching the binary, it would increase the time required to execute the attack. Similarly, during a debrief the red team explained that specifying options to the program through commandline parameters and a configuration file made it easier for them to understand what was running once they got access to a SCADA master replica; compiling these options into the program would make their information gathering more difficult.

B. Lessons for Successful Red Team and Deployment

Beyond the technical, we learned several lessons about how to prepare for and execute these types of experiments and deployments successfully. One of the key lessons was the importance of establishing rapport and personal connections well ahead of the actual experiment. Compared with three previous red team experiments that our group has taken part in over the years (with different systems), this one was somewhat less open and collaborative, which may have been partially due to the fact that the red team was physically located off-site. While this setup more realistically models the threat posed by remote cyberattacks, the ability to interact with the red team in person would have improved our ability to understand the exact attacks they carried out (which we mainly learned about from MANA, on-site monitoring, and a post-experiment debrief) and may have enabled them to launch a wider range of protocol-specific attacks once the low-level attacks had failed. The ability to work with power plant engineers in the second deployment experience led to a mutually beneficial experience that improved both sides' understanding of how our technology fits into the operation of a real power plant.

We also learned that the emerging open-source SCADA ecosystem has made it possible to effectively prepare for such experiments using emulation. Prior to the red team experiment, Spire had never been tested with a real PLC, but we were able to test the system end-to-end using OpenPLC [17] to emulate PLCs on Linux. This allowed us to set up the system in our lab and then transition to the real PLC in the deployment with only minimal changes.

Despite the ability to prepare via emulation ahead of time, both deployments required short-notice on-site development to make them a success. Both deployments were intensive two-week experiences, where the first week was devoted to setup, integration, and final development. These development and test environments were also significantly different from the normal environments we were used to, including a national lab with restricted Internet and phone use, as well as a power plant where protective personnel equipment (safety glasses, hard hats, steel-toed boots) was required at all times when moving between rooms.

C. Lessons for Transition in the Power Industry

The power plant deployment highlighted the complex nature of power grid monitoring and control systems today and the need for close collaboration with the power industry to develop holistic solutions. In contrast to a single monolithic SCADA system, the power plant included several distinct subsystems for generation and transmission. While our intrusion-tolerant SCADA system successfully controlled a small piece of a transmission topology, much more collaborative work is needed to understand the full power plant system architecture and design a holistic architecture for intrusion tolerance.

Moreover, given the conservative nature of the industry, an incremental transition approach is necessary to gain the trust of power grid operators and decision makers and mitigate concerns over introducing changes into systems where reliability is critical. The participation of the power plant engineers in the design of the red team experiment was essential in building their trust for the power plant deployment, and that deployment takes a first step toward building the confidence needed for ultimate transition, although this is likely to be a long process.

VII. Related Work

Previous work has used intrusion-tolerant replication to create SCADA systems that are resilient to SCADA master compromises and addressed many of the challenges of adapting real SCADA applications to fit the state machine replication model. However, to the best of our knowledge, none of these systems have been tested by a nation-state-level red team or deployed in a power plant.

Zhao et al. [4] use PBFT [10] with four replicated SCADA controllers and show that a simulated power grid scenario in a LAN can meet the required sub-second sampling rate of SCADA operations. Kirsch et al. [6] use Prime [11] to add intrusion tolerance to a Siemens SCADA product in a prototype implementation. Medeiros proposes a fault- and intrusion-tolerant architecture for the EDP Distribuicao SCADA system [5] that leverages knowledge of the Portuguese electric

grid to analyze the proposed architecture's ability to reduce downtime of SCADA services caused by benign faults.

Nogueira et al. [8] implement SMaRt-SCADA, an intrusion-tolerant prototype that integrates Eclipse NeoSCADA [18] with BFT-SMaRt [19]. The authors identify several challenges with making a traditional SCADA master support intrusion-tolerant replication, which they overcome by using proxies that serialize all messages and logical timeouts that synchronize timeouts across different replicas.

The BFT-SMaRt library underlying SMaRt-SCADA has also addressed the state transfer and state management challenges of supporting real applications that must tolerate partitions and proactive recoveries [20], allowing it to provide a BFT ordering service for the Hyperledger Fabric blockchain platform [21]. BFT-SMaRt provides a generic interface for persistent state management and transfer. In contrast, our approach is not generic but is tailored for SCADA systems where the state of field devices represents the ground-truth; this makes it possible to recover from temporary assumption breaches and does not require persistent state.

VIII. Conclusion

We have described our experience deploying intrusion-tolerant SCADA in a nation-state-level red team experiment and a power plant test deployment for the first time. These experiences offered technical lessons in supporting the assumptions of intrusion-tolerant protocols and employing defense-in-depth (integrating low-level security, protocol-level intrusion tolerance, and intrusion detection), as well as meta-lessons for experimentation and transition in the power industry.

Acknowledgement

We thank David Rolla, Bryan Tepper, John Tica, Keith Webster, and the rest of the team at the Hawaiian Electric Company for their strong support throughout this work. We thank Jim Brown, Cliff Eyre, David Linneman, and the rest of the team at Pacific Northwest National Lab for hosting and managing the red team experiment. This work was supported in part by DoD Environmental Security Technology Certification Program (ESTCP) Project EW-201607 to Resurgo LLC. Its contents are solely the responsibility of the authors and do not represent the official view of the Department of Defense.

References

[1] A. N. Bessani, P. Sousa, M. Correia, N. F. Neves, and P. Verissimo, "The crutial way of critical infrastructure protection," *IEEE Security & Privacy*, vol. 6, no. 6, pp. 44–51, Nov 2008.

[2] S. Zonouz, J. Rrushi, and S. McLaughlin, "Detecting industrial control malware using automated plc code analytics," *IEEE Security & Privacy*, vol. 12, no. 6, pp. 40–47, Nov 2014.

[3] D. Shelar, P. Sun, S. Amin, and S. Zonouz, "Compromising security of economic dispatch in power system operations," in *IEEE/IFIP Int. Conf. Dependable Systems and Networks (DSN)*, June 2017, pp. 531–542.

[4] W. Zhao and F. E. Villaseca, "Byzantine fault tolerance for electric power grid monitoring and control," in *Int. Conf. Embedded Software and Systems*, July 2008, pp. 129–135.

[5] N. A. C. Medeiros, "A fault- and intrusion- tolerant architecture for EDP distribuicao SCADA system," Master's thesis, University of Lisbon, 2011.

[6] J. Kirsch, S. Goose, Y. Amir, D. Wei, and P. Skare, "Survivable SCADA via intrusion-tolerant replication," *IEEE Trans. Smart Grid*, vol. 5, no. 1, pp. 60–70, Jan 2014.

[7] A. Babay, T. Tantillo, T. Aron, M. Platania, and Y. Amir, "Network-attack-resilient intrusion-tolerant scada for the power grid," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2018, pp. 255–266.

[8] A. Nogueira, M. Garcia, A. Bessani, and N. Neves, "On the challenges of building a bft scada," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2018, pp. 163–170.

[9] "Spire: Intrusion-tolerant scada for the power grid," http://www.dsn.jhu.edu/spire/, access: 2019-01-15.

[10] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.

[11] Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Prime: Byzantine replication under attack," *IEEE Trans. Dependable and Secure Computing*, vol. 8, no. 4, pp. 564–577, July 2011.

[12] A. Homescu, S. Neisius, P. Larsen, S. Brunthaler, and M. Franz, "Profile-guided automated software diversity," in *IEEE/ACM Int. Symp. Code Generation and Optimization (CGO)*, Feb 2013, pp. 1–11.

[13] Secure Systems Lab, "Multicompiler," https://github.com/securesystemslab/multicompiler, access: 2018-11-19.

[14] T. Roeder and F. B. Schneider, "Proactive obfuscation," *ACM Trans. Comput. Syst.*, vol. 28, no. 2, pp. 4:1–4:54, Jul. 2010.

[15] P. Sousa, A. Bessani, M. Correia, N. F. Neves, and P. Verissimo, "Highly available intrusion-tolerant services with proactive-reactive recovery," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 4, pp. 452–465, 2010.

[16] D. Obenshain, T. Tantillo, A. Babay, J. Schultz, A. Newell, M. E. Hoque, Y. Amir, and C. Nita-Rotaru, "Practical intrusion-tolerant networks," in *IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, June 2016, pp. 45–56.

[17] T. Alves, "The OpenPLC project," http://www.openplcproject.com/, access: 2018-12-05.

[18] "Eclipse neoscada," http://www.eclipse.org/eclipsescada/, access: 2018-12-04.

[19] A. Bessani, J. Sousa, and E. E. P. Alchieri, "State machine replication for the masses with bft-smart," in *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2014, pp. 355–362.

[20] A. N. Bessani, M. Santos, J. Felix, N. F. Neves, and M. Correia, "On the efficiency of durable state machine replication," in *USENIX Annual Technical Conference*, June 2013, pp. 169–180.

[21] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2018, pp. 51–58.