SAND2018-12221B

Moving Target Defense to Improve Industrial Control System Resiliency

Adrian R. Chavez

1. INTRODUCTION

Historically, control systems have primarily depended upon their isolation [1] from the Internet and from traditional Information Technology (IT) networks as a means of maintaining secure operation in the face of potential remote attacks over computer networks. However, these networks are incrementally being upgraded [2] and are becoming more interconnected with external networks so they can be effectively managed and configured remotely. Examples of control systems include the electric power grid, smart grid networks, micro grid networks, oil and natural gas refineries, water pipelines, and nuclear power plants. Given that these systems are becoming increasingly connected, computer security is an essential requirement as compromises can result in consequences that translate into physical actions [3] and significant economic impacts [4] that threaten public health and safety [5]. Moreover, because the potential consequences are so great and these systems are remotely accessible due to increased interconnectivity, they become attractive targets for adversaries to exploit via computer networks. Several examples of attacks on such systems that have received a significant amount of attention include the Stuxnet attack [6], the U.S.-Canadian blackout of 2003 [7], the Ukraine blackout in 2015 [8], and attacks that target control system data [9] itself. Improving the cybersecurity of electrical power grids is the focus of our research.

The power grid is responsible for providing electricity to society, including homes, businesses, and a variety of mission critical systems, such as hospitals, power plants, oil and gas refineries, water pipelines, financial systems, and government institutions. The “smart grid” acts as an advanced power grid with upgrades that provide power distribution systems and consumers with improved reliability, efficiency, and resiliency [10]. Some of the upgrades include automated energy load balancing, real-time energy usage tracking and control, real-time monitoring of grid-wide power conditions, distributed energy resources, advanced end devices with two-way communications, and improved processing capabilities. Advanced end devices, which are being integrated into smart grids, include programmable logic controllers (PLCs), remote telemetry units (RTUs), intelligent electronic devices (IEDs), and smart meters that are capable of controlling and performing physical actions, such as opening and closing valves, monitoring remote real-time energy loads, monitoring local events such as voltage readings, and providing two-way communications for monitoring and billing, respectively. These new devices replace legacy devices that have been in place for decades that were not originally designed with security in mind since they were previously closed systems without external network connectivity. Although these new devices aid in efficiency, they may create more avenues for attack from external sources.

Additionally, control systems are often statically configured [11] over long periods of time and have predictable communication patterns [12]. After installation, control systems are often not replaced for decades. The static nature combined with remote accessibility of these systems creates

an environment in which an adversary is well positioned to plan, craft, test, and launch new attacks. Given that the power grid is actively being developed and advanced, the opportunity to incorporate novel security protections directly into the design phase of these systems is available and necessary. Of particular interest are defenses that can better avoid both damage and loss of availability, as previously documented in the power grid [5], to create a more resilient system during a remote attack over computer networks.

1.1. Challenges

One of the main challenges of introducing modern computer security protections into Industrial Control Systems (ICSs) is to ensure that the computer security protections themselves not only improve the security of the overall system, but also do not impede the operational system from functioning as expected. A security solution that is usable and practical within an IT environment may not necessarily be practical within an ICS environment. ICSs often have real-time requirements and any newly introduced software or security solution must also meet those same requirements.

Another challenge is to identify useful metrics that quantify the effectiveness of the Moving Target Defense (MTD) techniques from the perspective of both the adversary and the defender of the system. The goal of the adversary is to exploit the system before the MTD defends against the attack by modifying the environment. The goal of the defender is to change the environment frequently enough to evade an adversary, but not too frequently so that system performance is negatively impacted. Finding the correct balance is necessary so that the adversary cannot exploit the system and the MTD strategy does not prevent the system from maintaining a normal operating state.

Gaining access to representative ICS environments is another challenge when developing new security protections for ICS systems. Modeling and simulation tools can be effective, but gaining a true understanding of the consequences and effects of deploying a new security protection in practice requires validation within a representative ICS environment. Several factors, such as network load, processor load, and memory load are difficult to accurately project within a simulated environment. The harsh working conditions of ICSs (such as wide temperature ranges) are one element to consider when deploying new technologies within these environments.

1.2. MTD within Critical Infrastructure

Critical infrastructure systems bring in a distinctive set of constraints and requirements when compared against traditional IT-based systems. Critical infrastructure systems are often time-sensitive with stringent real-time constraints, as is the case for cyber-physical systems [13]. It is therefore important for any new computer security protections introduced to also meet these same time requirements so that they do not negatively affect the operational network. Additionally, the most important requirements for these systems are often to maintain high availability and integrity due to the nature of the systems that they control (e.g., the electrical power grid, water pipelines, oil and natural gas refineries, hospitals, residential and commercial buildings, etc.). Any loss of availability can result in significant consequences not only in terms of economics, but also in terms of public health and safety. Similarly, compromising the integrity of these systems, such as sending maliciously modified commands, can result in similar consequences. Also of note, is that critical infrastructure systems are composed of both legacy and modern systems that must interoperate with one another without affecting availability and security. New security solutions must therefore

take interoperability into account so that they can scale without the requirement of upgrading every end device within the system. Additionally, in order for MTD-based approaches to be successfully deployed within critical infrastructure environments, they must satisfy the distinct time constraints and requirements of these environments. Since the time constraints vary from one system to another, the stricter time requirement used for teleprotection systems should be used (12-20 ms) [14]. For Supervisory Control and Data Acquisition (SCADA) communications, those requirements can, in some cases, be relaxed to 2-15 seconds or more.

2. BACKGROUND

Artificial diversity is an active area of research with the goal of defending computer systems from remote attacks over computer networks. Artificial diversity within computer systems was initially inspired by the ability of the human body's natural immune system [15] to defend against viruses through diversity [13]. Introducing artificial diversity into the Internet Protocol (IP) layer has been demonstrated to work within a software-defined network (SDN) environment [16]. Flows, based on incoming port, outgoing port, incoming media access control (MAC) and outgoing MAC, are introduced into software-defined switches from a controller system. The flows contain matching rules for each packet and are specified within the flow parameters. If a match is made within a packet, then the flow action is to rewrite source and destination IP addresses to random values. The packets are rewritten dynamically while they are in flight traversing each of the software-defined switches. Although applying artificial diversity on an SDN has been demonstrated, the effectiveness of such approaches has not been quantitatively measured. Furthermore, to our knowledge, the approach has not been deployed within an ICS setting, which differs substantially from traditional IT-based systems.

It has also been demonstrated that IP randomization can be implemented through the Dynamic Host Configuration Protocol (DHCP) service that is responsible for automatically assigning IP addresses to hosts within the network [17]. Minor configuration modifications to the DHCP service can be made to specify the duration of each host's IP lease expiration time to effectively change IP addresses at user-defined randomization intervals. However, this approach only considers long-lived Transmission Control Protocol (TCP) connections, otherwise disruptions in service will occur as the TCP connection will need to be re-established. Service interruptions within an ICS setting is not an option due to their high availability requirements. Quantifying the effectiveness of such approaches has also not been performed within an ICS setting outside of surveys [18] that evaluate MTD techniques within an IT setting, where IP randomization by itself was qualitatively ranked to have low-effectiveness with low operational costs. Also of note is that IP randomization approaches by themselves have been demonstrated to be defeated through traffic analysis where endpoints of the communication stream can be learned by a passive adversary observing and correlating traffic to individual endpoints [19].

Anonymization of network traffic is an active area of research with several implementations available in both commercial and open source communities. MTD and anonymization are related in that they both have the goal of protecting attributes of a system from being discovered or understood. One of the early pioneering groups of anonymous communications describes the idea of onion routing [20], which is widely used today. This approach depends on the use of an overlay network made up of onion routers. The onion routers are responsible for cryptographically removing each layer of a packet, one at a time, to determine the next hop routing information to eventually forward each packet to their final destinations. The weaknesses of this solution are that

side channel attacks exist and have been demonstrated to be susceptible to timing attacks [21], packet counting attacks [22], and intersection attacks [23], which can all reveal the source and destination nodes of a communication stream.

The Onion Router (Tor) is one of the most popular and widely used implementations of onion routing with over 2.25 million users [24]. Tor is able to hide servers, hide services, operate over TCP, anonymize web-browsing sessions, and is compatible with Socket Secure (SOCKS)-based applications for secure communications between onion routers. However, it has been shown empirically with the aid of NetFlow data that Tor traffic can be de-anonymized with accuracy rates of 81.4% [25]. The results are achieved by correlating traffic between entry and exit points within the Tor network to determine the endpoints in communication with one another. Furthermore, Tor has an overhead associated with the requirement to encrypt traffic at each of the onion routers; this overhead would need to be limited within an ICS environment to meet the real-time constraints required of these systems. Similarly, garlic routing [26] combines and anonymizes multiple messages in a single packet, but is also susceptible to the same attacks.

Overlay networks have similar features as Tor, but with the goal of reducing the overhead associated with a Tor network. It has been shown that overlay networks can be used to mitigate Distributed Denial of Service (DDoS) attacks [27]. The overlay networks reroute traffic through a series of hops that change over time to prevent traffic analysis. In order for users to connect to the secure overlay network, they must first know and communicate with the secure overlay access points within the network. The required knowledge of the overlay systems prevents external adversaries from attacking end hosts on the network directly. This design can be improved by relaxing the requirement of hiding the secure overlay access points within the network from the adversary. If an adversary is able to obtain the locations of the overlay access points, then the security of this implementation breaks down and is no longer effective.

Steganography is typically used to hide and covertly communicate information between multiple parties within a network. The methods described in current literature [28] include the use of IP version 4 (IPv4) header fields and reordering IPsec packets to transmit information covertly. Although the focus of the steganography research is not on anonymizing endpoints, it can be used to pass control information to aid in anonymizing network traffic. The described approach would have to be refined to increase the amount of information (e.g., $log_2(n!)$ bits can be communicated through n packets) that can be covertly communicated if significant information is desired to be exchanged. Steganography techniques have the potential to facilitate covert communication channels for MTD techniques to operate correctly, but have not been applied in this fashion.

Transparently anonymizing IP-based network traffic is a promising solution that leverages Virtual Private Networks (VPNs) and the Tor [29] service. The Tor service hides a user's true IP address by making use of a Virtual Anonymous Network (VAN), while the VPN provides the anonymous IP addresses. The challenge of this solution is the requirement that every host must possess client-side software and have a VPN cryptographic key installed. In practice, it would be infeasible for this approach to scale widely, especially within ICS environments where systems cannot afford any downtime to install and maintain the VPN client-side software and the cryptographic keys that would be necessary at each of the end devices. To reduce the burden on larger scale networks, it may be more effective to integrate this approach into the network level, as opposed to at every end device, using an SDN-based approach. In order for an MTD strategy to be effective within an ICS

environment, the MTD solution must have the ability to scale to a large number of devices without significant interruptions in communications.

2.1. MTD Techniques

MTD is an active area of research that seeks to thwart attacks by invalidating knowledge that an adversary must possess to mount an effective attack against a vulnerable target [30]. For each MTD defense deployed, there is an associated delay imposed on both the adversary and on the defender of the operational system. Quantitatively analyzing the delays introduced by each additional MTD technique applied individually and in combination with one another within an ICS environment is necessary before deploying an MTD strategy. Our analysis will aid in optimally assigning the appropriate MTD techniques to enhance the overall security of a system by minimizing the operational impacts while maximizing the adversarial workload to a system. We also evaluate each of the possible MTD defenses placed at various levels of an ICS system.

2.2. MTD Categories

Because power systems are statically configured and often do not change over long periods of time, those environments are ideal for introducing and evaluating MTD-based protections. The goal of the various MTD techniques presented here is to increase the adversarial workload and level of uncertainty during the reconnaissance phases of an attack. Since it remains an open problem to completely stop a determined, well-funded, patient, and sophisticated adversary, increasing the delay and likelihood of detection can be an effective means of computer security. There are a variety of MTD approaches that can be categorized according to where the defense is meant to be applied, including at the application level, the physical level, or the data level of a system. Five high-level MTD categories have been described as part of a MTD survey [31], which include dynamic platforms, dynamic runtime environments, dynamic networks, dynamic data, and dynamic software. These categories are described in the sections that follow within the context of a critical infrastructure environment.

2.2.1. Dynamic Platforms

PLCs, RTUs, and IEDs vary widely from one site to another within an ICS environment. There are a number of vendors that produce these end devices with different processor, memory, and communications capabilities. These devices are responsible for measuring readings from the field (such as power usage within a power grid context) and taking physical actions on a system (such as opening or closing breakers in a power grid). Many of these end devices are several decades old and must all be configured to work together. If an adversary has the ability to exploit and control these types of end devices, they would have the ability to control physical actions remotely through an attack over computer networks. At the physical layer, several MTD strategies exist to increase the difficulty of an adversary's workload to successfully exploit a system. One strategy rotates the physical devices that are activated within a system [32, 33, 34]. For this strategy to work, the physical devices and software may vary widely, but the only requirement is that they must be capable of taking in the same input and successfully producing the same output as the other devices. If there are variations in the output, then alerts can be generated to take an appropriate action. These approaches increase the difficulty from an adversary's perspective because the adversary would be required to simultaneously exploit many devices based on the same input instead of exploiting just a single device. The difficulty for the defender comes in the form of having additional devices that must be administered and managed while also ensuring the security

of the monitoring agents is maintained and that they do not become additional targets themselves. Strategies such as n-variant [35] MTD techniques run several implementations of a particular algorithm with the same input where variations of outputs would be detected by a monitoring agent. Others [36] have shown firmware diversity in smart meters can limit the effectiveness of single attacks that are able to exploit a large number of devices with a single exploit. Customized exploits would have to be designed specifically for each individual device.

2.2.2. Dynamic Runtime Environments

Instruction Set Randomization (ISR) [37] and Address Space Layout Randomization (ASLR) [38] are MTD techniques that modify the execution environment of an application over time. The effectiveness of such techniques has been measured in traditional enterprise networks, but has not yet been measured on devices found within ICS-based environments where real-time responses are a major requirement. The impact upon the real-time response requirement has been measured along with the adversary's increased workload when ISR [39] and ASLR [38, 40] are enabled.

2.2.3. Dynamic Networks

Section 2 describes many of the network randomization research efforts that have been performed. The resilience of network MTD techniques against several adversaries with different capabilities has been measured in prior work [41]. For network-based MTD techniques to be effective, the exact point at which the benefit of each MTD strategy to the defender is maximized and the adversarial workload is maximized must be found. The analysis should also take into account that ICS systems have strict real-time and high availability constraints. The MTD parameters used, such as rates of randomization and the location of the MTD techniques themselves (at the network level or the end device level), to find the balance between security and usability should ensure that the solution does not hinder the operational network.

2.2.4. Dynamic Data

Randomizing the data within a program is another technique used to protect data stored within memory from being tampered with or exfiltrated [42]. Compiler techniques to xor memory contents with unique keys per data structure [43], randomizing Application Programming Interfaces (APIs) for an application, and Structured Query Language (SQL) string randomization [44] help protect against code injection type attacks. These techniques have been demonstrated on web servers and have shown varying levels of impacts to the operational systems. The benefits are that adversaries can be detected if the data being randomized is accessed improperly when the system is being probed or an attack is being launched in the case of SQL string randomization.

The same techniques can be applied and measured within a control system environment to assess the feasibility of applying such techniques and meeting the real-time constraints. For example, a historian server typically maintains a database of logs within an ICS environment. This server is a prime location to apply SQL string randomization towards database accesses. Data randomization can be performed on the data stored within the registers of a PLC. To measure the effectiveness of this technique, metrics of the response times to find the delays introduced can be captured. After gathering these measurements, an evaluation of the delays introduced can be performed to ensure that they are within the acceptable limits of an ICS environment. These are a few examples of where data randomization can be applied within an ICS setting.

2.2.5. Dynamic Software

Introducing diversity into software implementations helps eliminate targeted attacks on specific versions of software that may be widely distributed and deployed. In the case of a widely deployed software package, compromising a single instance would then compromise the larger population of deployed instances. To introduce diversity and help prevent code injection attacks in networks, the network can be mapped to a graph coloring problem [45] where no two adjacent nodes share the same color or software implementations. This type of deployment helps prevent worms from spreading and rapidly infecting other systems in a network using a single payload. These techniques should also be considered as a defensive mechanism within an ICS environment. However, metrics and measurements need to be gathered and evaluated using software that is deployed and found within operational ICS environments.

At the instruction level, metamorphic code is another strategy that has primarily been utilized by adversaries to evade anti-virus detection [46]. The code is structured so that it can modify itself continuously in time and maintain the same semantic behavior while mutating the underlying instructions of the code. The idea is similar to a quine [47] where a program is capable of reproducing itself as output. Metamorphic code reproduces semantically equivalent functionality but with an entirely new and different implementation with each replication. There are many techniques to develop metamorphic code-generating engines, but they are typically not used as a defensive strategy.

When software remains static, it becomes a dependable target that can be analyzed, tested, and exploited over long periods of time by an adversary. Introducing diversity at the instruction level helps eliminate code injection attacks, buffer overflows, and limits the effectiveness of malware to a specific version of software in time. Once the code self-modifies itself, the malware may no longer be effective, depending on the self-modification being performed. Several techniques [48] exist to use self-modifying code as a defensive mechanism, such as inserting dead code, switching stack directions, substituting in equivalent but different instructions, in-lining code fragments, randomizing register allocation schemes, performing function factoring, introducing conditional jump chaining, enabling anti-debugging, and implementing function parameter reordering.

2.2.6. Dead Code

Dead code refers to function calls to code fragments that do not contribute to the overall goal of an algorithm and is a useful strategy to deter an adversary. Dead code fragments have the goal of causing frustration, confusion, and generally wasting the time of an adversary in analyzing complex code fragments that are not of importance to the overall algorithm. However, techniques do exist to dynamically detect dead code [49] fragments, so this strategy should be deployed with care. Also of note is that if the size of a program cannot exceed a certain threshold, it may be necessary to take into account the available space on the system so that the code does not overly cause an excess amount of bloat and exceed space limitations.

Dead code can help protect against an adversary who is statically analyzing and reverse-engineering a software implementation. In this case of a MTD protection, when the dead code is included, the goal is to cause the adversary to spend a significant amount of time analyzing code that is not useful to the overall software suite. This technique serves as a deterrence and a decoy to protect the important software. Dead code is often used as an obfuscation technique of software to make it more difficult for an adversary to understand [50].

Although this technique may be effective against certain types of adversaries performing static code analysis, the security is based on the assumed limited analytical and intelligence capabilities placed on an adversary. This assumption is not valid when considering sophisticated adversaries who have a wide array of resources in terms of finances, staff, and intelligence available. The technique also breaks down and fails when dynamic code analysis is performed to recognize that the dead code does not actually provide any contributions to the overall functionality of the software under consideration.

2.2.7. Stack Directions

The direction that the stack grows can be chosen to grow either at increasing memory addresses or decreasing memory addresses [51]. Buffer overflow attacks must take into account this direction to effectively overflow the return address so that the adversary can execute their own arbitrary code. One strategy to eliminate such an attack is to either run a program that dynamically selects the direction of the stack at runtime or to run two instances of a program in parallel with each instance having their stacks growing in opposite directions. The two programs would then be overseen by a monitoring agent to ensure that there are no deviations in results between the two programs. It is possible that an attack can still succeed in both cases at the same time, but only in very specialized cases where the original code is written in such a way that the overflow works on different variables simultaneously in both directions that the stack grows; this is, however, unlikely to occur on the majority of code that is of practical interest to an adversary.

This technique must consider possible space limitations of the system and the overhead to detect deviations between the two versions of software. If both versions are running on the same system, then the processor utilization may also be a concern for other applications running on the system. If the implementations are on separate systems, then the network overhead to communicate the results of each run must also not negatively impact the system in question. An additional area of importance is the security of the monitoring agent to validate that a possible attack is in progress. Many security protections often become a target [52] for adversaries and need to be taken into account, as well.

2.2.8. Equivalent Instruction Substitution

Many techniques exist to introduce diversity into a program by substituting equivalent instructions [53]. The goal of substituting equivalent instructions for multiple instances of a program is to maintain equivalent functionality while diversifying the implementation of the underlying software. The benefit is that the difficulty of identifying functionally equivalent software implementations from one another is increased from the adversaries' perspective. This increases the difficulty placed on an adversary who is attempting to develop a scalable malware solution designed to compromise a large number of systems using a single exploit. The tradeoff is typically in the increased performance of the variants of equivalent instructions. In many cases, compiler optimizers will automatically reduce high-level programming modifications to the same optimized assembly instructions using dependency graphs [54]. To disable compiler optimizations, compiler flags must be enabled to maintain the intended diversity within the binary executable. The impact on the defender and operational network of applying this approach has not been measured within an ICS environment at the time of this writing. The feasibility of applying this approach within an ICS setting will depend on the availability and number of equivalent instructions that can be replaced that function the same and do not introduce significant time delays.

3. MTD APPLICATIONS AND SCENARIOS WITHIN ICS

MTD strategies can benefit a broad range of environments spanning enterprise IT systems that are widely connected and ICS networks, which are completely isolated from the Internet. Each environment has different requirements and constraints for which the MTD approaches and parameters must be specifically configured in order for the strategy to be feasible in a practical setting. Some of the MTD parameters that can be adjusted include the frequency of reconfigurations, the amount of entropy supplied to the MTD technique when performing IP address randomization, the maximum number of hops between endpoints tolerable when performing communication path randomization, and the size of a binary when performing application randomization, for example. The requirements and constraints of these systems include meeting strict performance measurements (e.g., latency, bandwidth, throughput constraints), satisfying the North American Electric Reliability Corporation (NERC) Critical Infrastructure Protection (CIP) Standards [55], the International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 27000 series of Information Security Management Systems (ISMS) standards [56], and conformance to the National Institute of Standards and Technology (NIST) Cybersecurity Framework [57].

Each environment has its own unique set of requirements and constraints that must be met in an operational setting. Because MTD approaches can be applied broadly across a number of environments, the parameters of the MTD strategies must be adjusted to meet the requirements and constraints of the target environment. The focus here is on ICS environments, but the approaches can be applied similarly to other environments including enterprise networks, Internet of Things (IoT), cloud, mobile, etc.

3.1. Industrial Control Systems

The primary requirement for many of these systems is to maintain high availability and integrity [1]. In the electrical power grid, the high availability requirement comes from the criticality of the types of systems that depend on the power grid to operate (e.g., hospitals, governments, educational institutions, commercial and residential buildings, etc.). Figure 1 shows an example power grid and the components found at various layers of the network. These systems involve a number of utilities communicating with one another and the distribution of power across a geographically disperse area of customers. A study was performed with the goal of quantifying the economic costs associated with service interruptions to the U.S. power grid and are estimated to be approximately $79 million annually [58]. From the 2003 blackout in New York [59], the estimated direct costs were between $4 billion and $20 billion [60] while there were also in excess of 90 deaths [61]. Though these interruptions were not due to remote attacks over computer networks, such attacks are capable of causing similar disruptions. The need for computer security within an ICS setting is clear as the impacts and consequences of downtime can be dire.

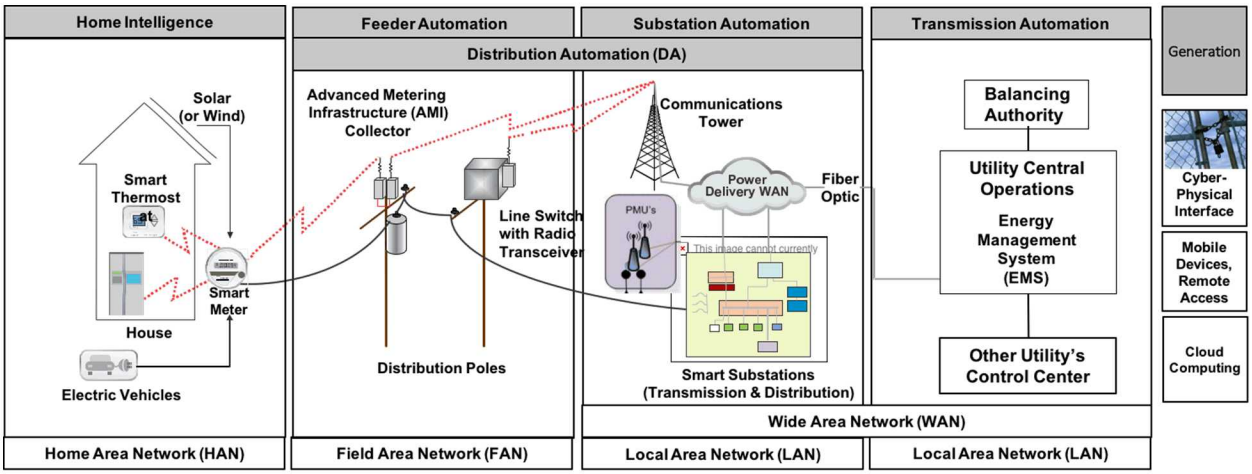

Figure 1. An example power grid that shows the high-level components from generation of power to transmission to distribution and finally to delivery at a residential home.

3.1.1. Use Case

Because ICSs operate with both legacy and modern devices, there is a mixture of serial and IP communications. Typical protocols deployed within ICS networks include Modbus [62], Distributed Network Protocol (DNP3) [63], and Process Field Net (PROFINET) [64]. These protocols are widely used within ICS environments and many, such as Modbus, were not designed with security in mind since these protocols were originally intended for serial communications, and only later expanded, with Modbus TCP, to function over IP networks. Still, the expectation was that such IP networks would be controlled and isolated. Modbus is a protocol that can be used to read and write memory values to ICS end devices, such as PLCs or industrial computers that can either sense readings from equipment or perform physical actions based on digital inputs received. Some of the physical actions include opening or closing a valve within a water pipeline, opening or closing breakers within a power system, or shutting down a power plant.

Given that ICSs are becoming more interconnected to business networks for ease of maintenance and management, remote attacks over computer networks become a real possibility since the business networks are connected to the Internet. However, as demonstrated by Stuxnet [6], a network connection to the Internet is not a requirement to exploit a system, and the attack against Home Depot [65] shows how vulnerable operational technology can be exploited to penetrate additional systems. In a scenario where the Modbus protocol is configured to read and write memory values from and to, respectively, a PLC that controls a physical process, an adversary could launch a man-in-the-middle (MITM) attack [66] to spoof values read/written to the PLC's memory. Since legacy PLCs are fundamentally different from the systems we are accustomed to working with (in terms of the memory and processing resources available) and because they were designed with the understanding that they would be used only within closed system environments, integrity and authentication checks were typically not built into these systems. As ICS environments have evolved, PLCs and other end devices are becoming more connected externally. As a result, end devices that do not have integrity and authentication checks built in are susceptible to adversaries eavesdropping on communications and/or maliciously modifying those communications via MITM attacks. To mitigate such an attack, a defender could deploy a number of strategies to protect against this threat.

If the adversary has direct access to the network and has the ability to observe or modify traffic, spoofed packets can be injected or replayed into the network. The goal of the adversary in this scenario would be to maliciously write incorrect values into a PLC's memory space to cause an unintended physical action to take place within the system. One defense that could protect against an attack where the adversary crafts and injects packets into the network could be to deploy an MTD strategy randomizing application port numbers in the communication channel (the Modbus standard port number is 502). Continuously changing this value in time would require the adversary to constantly track and learn the new random mappings that are active. Another defense that can be deployed would be to configure a secure communication channel between the endpoints to prevent the adversary from maliciously observing and spoofing traffic. This solution would require the adversary to compromise the underlying encryption algorithm or a cryptographic key.

The optimal solution that a defender should select depends on the capabilities of the end devices, as well as the amount of delay that can be tolerated by the network. If the end devices are capable of supporting more well-established modern encryption algorithms, such as the Advanced Encryption Standard with at least a 128-bit key length (AES-128) [67], then that is the ideal solution. However, the end devices may either not be capable of supporting AES or they may not be able to afford the computationally expensive tasks, in terms of central processing unit (CPU) utilization [68], to support an encrypted channel. The amount of CPU available depends on the current load of the system. The other option is to deploy a gateway system capable of serving as a proxy to harness the necessary security protections [69, 70, 71, 72]. This MTD approach follows the gateway solution and is capable of minimally delaying the network communications, while adding on an additional layer of defense into the network. The parameters of the MTD techniques can then be adjusted to meet the criteria required by the ICS to maintain a high availability system, while avoiding the computationally expensive price of encrypting all communication channels.

3.1.2. Constraints

One of the major challenges for new technologies to be deployed within ICS environments is that legacy and embedded devices occupy a large portion of these systems. Some of the devices found are decades old and do not have the processing or memory resources available to harness modern security technologies. This can be attributed to the fact that many of these systems were developed starting from the 1880's to the 1930's [73] and many legacy devices are still in place today. Another constraint is that, even if the devices are modern and capable of harnessing new security technologies, the software and specialized hardware are often both closed and proprietary [1]. The proprietary nature creates a challenge for security researchers to understand, integrate, and test new security protections directly into the end devices themselves. In this scenario, an additional gateway system is typically introduced to proxy the end devices with the new security technologies enabled. This proxy creates an additional hop that packets must traverse, which affects latency.

Another challenge is the diverse set of equipment that can be found within ICS environments. These devices, from multiple vendors, must interoperate with one another, which is a challenge of its own. Adding computer security protections into each of these devices directly in a vendor neutral way requires agreement and collaboration between a number of competing parties. This is a challenge that can oftentimes be the most difficult piece of the puzzle. These constraints cannot be ignored as new security technologies must be retrofitted into the existing environment with competing vendors working together, as completely replacing all of the equipment is not a valid option.

3.1.3. Requirements

ICSs have several requirements, regulations, and standards that must be met. Perhaps the most important requirements for ICS environments are to minimize the amount of delay introduced into a system and to ensure the integrity of the commands communicated within these environments. Latency is one of the primary metrics used and is typically constrained to 50 milliseconds and in some cases can be in the nanosecond [74] scale. Any delays on the operational network can result in instability of the power system [75]; therefore, new security protections must meet the strict time requirements to be relevant and feasible within these systems. Integrity is also a key requirement as data integrity attacks could manipulate sensors, control signals, or mislead an operator into making a wrong decision [9]. Also, interoperability requirements, as mentioned in the preceding subsection, must be met. The International Electrotechnical Commission 61850 (IEC-61850) [76] standard has outlined a general guide to achieve interoperability. To maximize the benefit of new security features introduced into an ICS system, these requirements and standards need to be met.

4. EXPERIMENTATION

We evaluated network-based moving target defense strategies applied to a representative control system environment. The environment consisted of nine buildings with inverters communicating with a central server. The system evaluated will harness a 2.5 megawatt microgrid system and our tests were performed before end devices were introduced into the network to evaluate the MTD strategies independently. Figure 2 shows the configuration of the network where IP randomization and port randomization were integrated. These MTD strategies continuously modified IP addresses and port numbers at varying frequencies from one second to completely static configurations. Our testing involved three buildings where we were able to successfully demonstrate the MTD technologies. Prior to introducing the MTD technologies, each of the three buildings included a Software Defined Networking (SDN) capable switch to manage the MTD techniques. The three SDN capable switches are shown in Figure 2 and labeled as “SDN Switch Flow Forwarding” in the lighter blue rectangles in each of the buildings. An end device was included in Building #1 labeled as “Host 1” to represent an end device within the network. An additional Linksys router was added to the network to support an out-of-band communication channel for the SDN OpenFlow control traffic. The dashed green lines show the out-of-band communications.

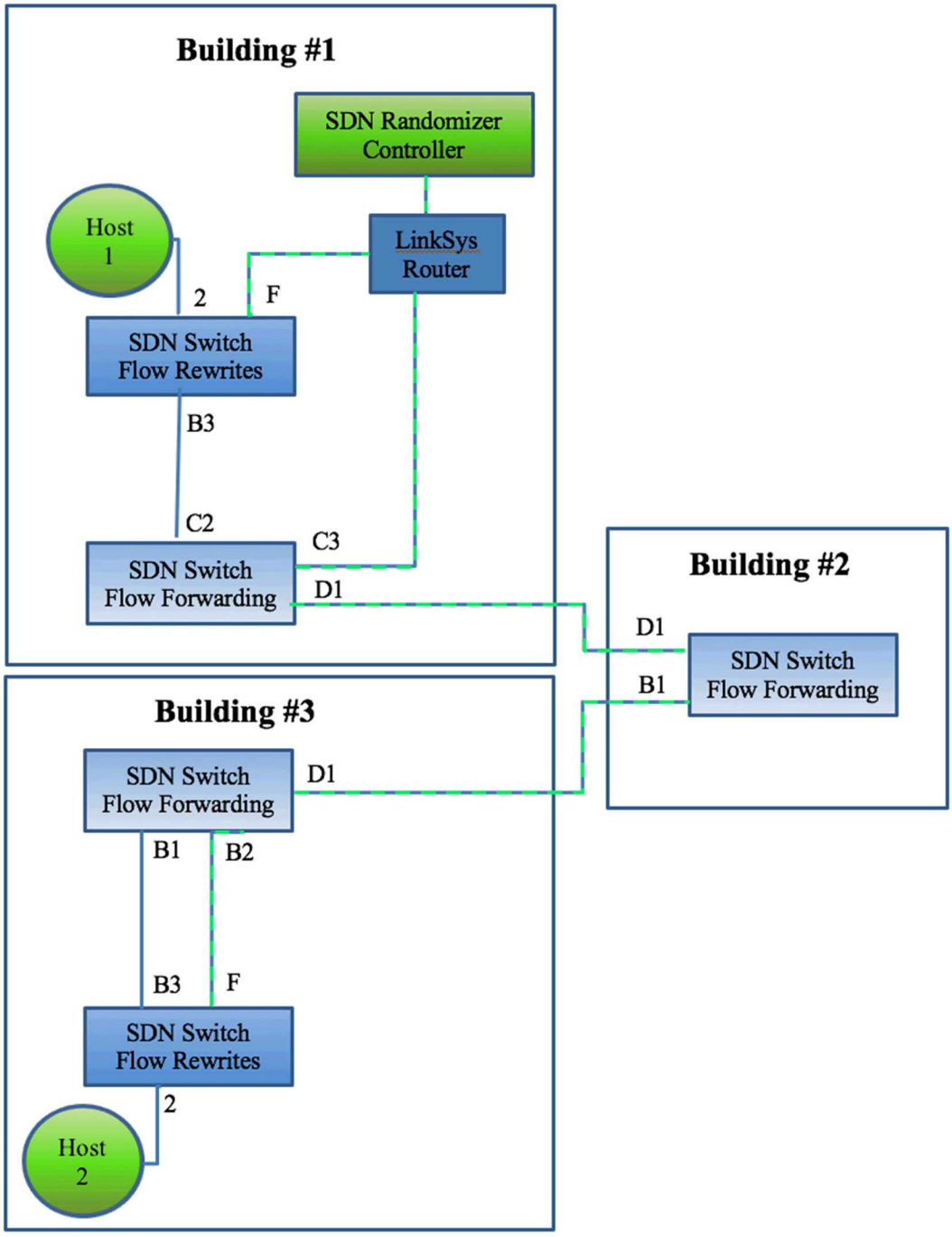

Figure 2. Multi-building SDN configuration with IP randomization enabled.

To manage the randomized flows installed on the SDN switches, the "SDN Randomizer Controller" was also added to the network. This system is responsible for periodically communicating the randomized IP addresses to each of the SDN switches to create the moving target defense solution. The same "SDN Switch Flow Rewrites" SDN-capable switch was also introduced into Building #3. Additionally, a second host, labeled "Host 2", was added to Building #3 to show communications between two end points within the network. Building #2

only has an SDN capable switch, labeled "SDN Switch Flow Forwarding," which has flows configured to forward both control and data traffic between Building #1 and Building #3.

"Host 1" and "Host 2" are then able to communicate with one another with IP randomization enabled. Additionally, "Host 1" and "Host 2" have machine-learning dynamic defense algorithms enabled to detect anomalous behavior on the network. Flows with the source IP address of the "SDN Randomizer Controller" are identified as management communications and are forwarded to the correct ports of the "SDN Switch Flow Rewrites," which are both shown as port "F" in the diagram.

4.1. Adversarial Scenario

We showed that if an adversary were to be introduced into the network either as: (1) an insider; or as (2) a result of a successfully launched MITM attack anywhere between the two "SDN Switch Flow Rewrites," then we could detect a network scan and automatically respond by randomizing the IP addresses to invalidate the information gained from the network scan. In our specific scenario, the adversary was introduced in building #1 in the link with "B3" and "C2" as the end point ports. The adversary then launched the following nmap scan:

```
nmap -sP x.y.z.0/24
```

where x.y.z is the network address configured in the system. The network in this case was configured as a 24-bit network mask with the last 8-bits reserved for host IP addresses. The nmap command above will scan the entire IP address space of the x.y.z network and report back the hosts that are active. In this scenario, once the adversary receives the results, they would then attempt to open a secure shell (ssh) session with the hosts in the network. The goal of the MTD technology is to detect the initial network scan and randomize IP addresses so that information gained about active hosts within the network would no longer be valid. To accomplish this goal, we launched the MTD strategy upon detection of the network scan. The described use case was successfully deployed within our testbed.

4.2. Metrics

Metrics were captured when the randomization schemes were performed independently and when combined with one another. The metrics collected include round trip time, bandwidth, throughput, TCP retransmits, and dropped packets. The randomization schemes evaluated include a baseline measurement without any randomization schemes enabled, IP address randomization with a frequency interval of three seconds, port randomization with a frequency interval of three seconds, and port randomization with a frequency interval of one second with encryption. The results are shown in Figure 3 through Figure 8.

Figure 3 and Figure 4 show the results of the Internet Control Message Protocol (ICMP) round trip times measured using the OpenDaylight controller as applied towards the scenario, as shown in Figure 2, where "Host 1" is issuing the following command to "Host 2":

```
ping -c 300 host2.
```

In Figure 3, each of the randomization schemes are measured independently and when combined with one another. The impacts vary slightly and are within the noise of network traffic as each scheme fluctuates outperforming and underperforming the other schemes depending on when the

measurement is taken. This can be seen more clearly in Figure 4 where the raw averages and standard deviations closely resemble one another across each of the randomization schemes. The impacts of each of the randomization schemes in our test environment proved to be minimal from our experimental results obtained.

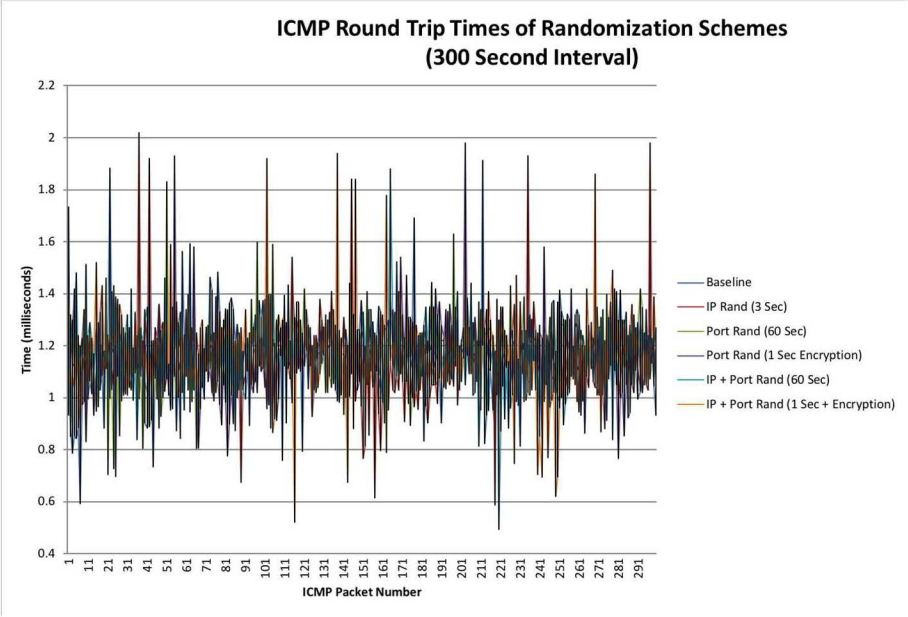

Figure 3. ***ICMP Round Trip Time measurements when enabling each randomization scheme independently and when combined with one another over a 300 second interval.***

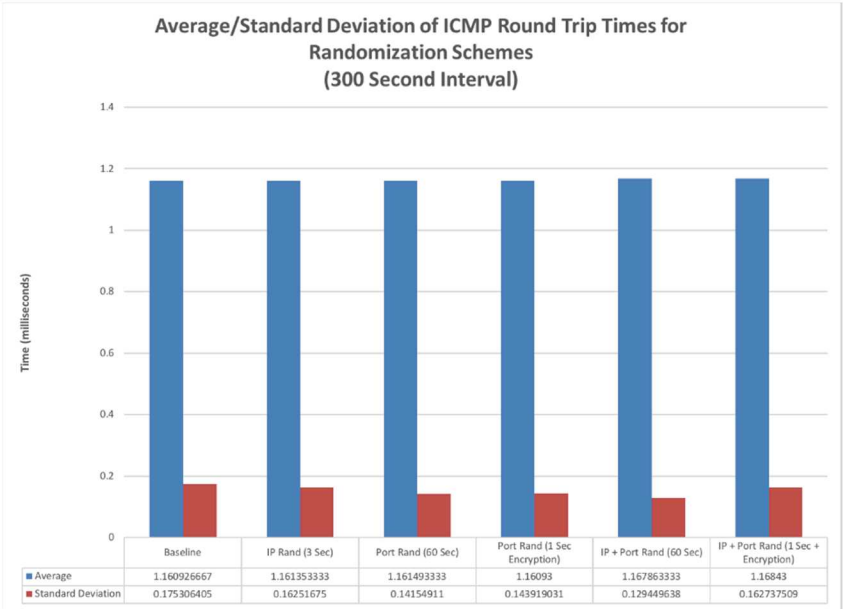

Figure 4. ***The averages and standard deviation for the round trip times when enabling each of the randomization schemes independently and in combination of each other over a 300 second interval of time.***

Figure 5 and Figure 6 measure the effects that each of the randomization schemes have on the transfer rates and the bandwidth. Since we were working with 100 megabits per second links, the maximum amount of data that can be transferred is 12.5 megabytes (100/8). Our results show that

most of the schemes, including the baseline, achieve ~11.2 megabytes of data transferred. The exceptions to this are the schemes that use the port randomization with encryption involved where ports are randomized and encrypted every second. Since we are randomizing at each of the endpoints in software and the cost for AES encryption is significant, the transfer rates were significantly impacted and yielded ~0.1 megabytes of data transferred. These results indicate that in environments where large amounts of data need to be transferred quickly, that encrypting and randomizing ports every second may not be a suitable option. However, the impacts on round-trip time were minimal so if small amounts of data are transferred, as is typically the case within control system communications, then any of the schemes may be appropriate. The metrics captured for transfer rates and bandwidth were captured with the following command:

```
iperf3 -c host2 -i 1 -t 300 -p 999 -V.
```

This command was issued for the client to connect to "Host 2" on port 999 and report back results every second over a 300 second total interval in verbose mode.

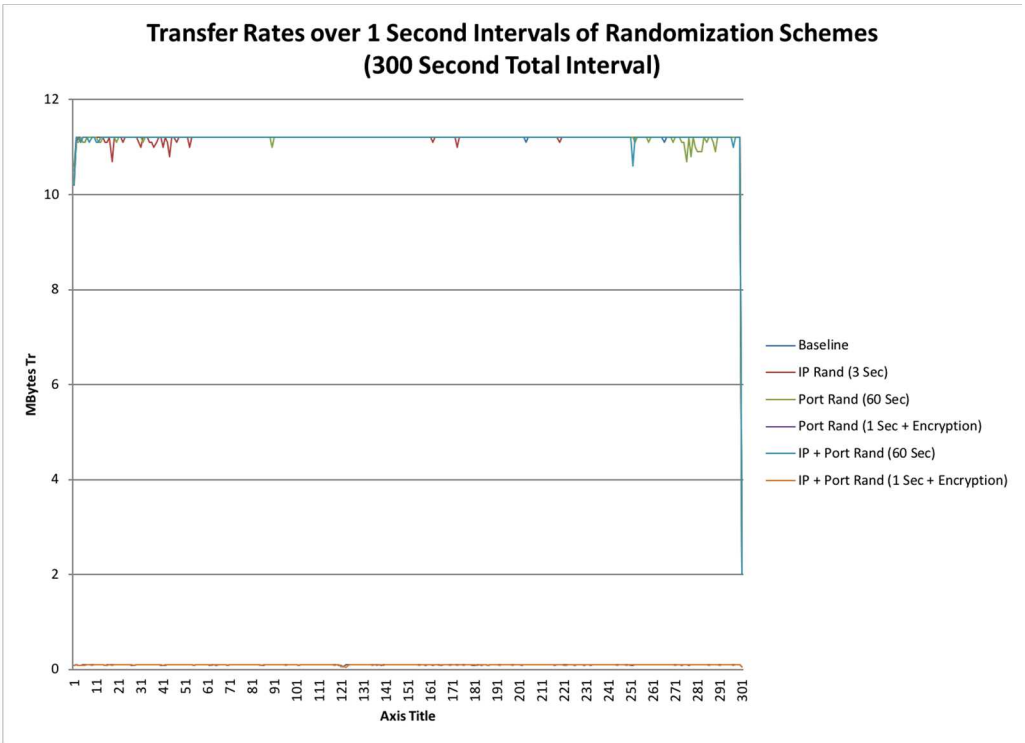

Figure 5. ***The amount of data transferred when enabling each randomization scheme independently and in combination of each other over one second increments in a 300 second interval of time.***

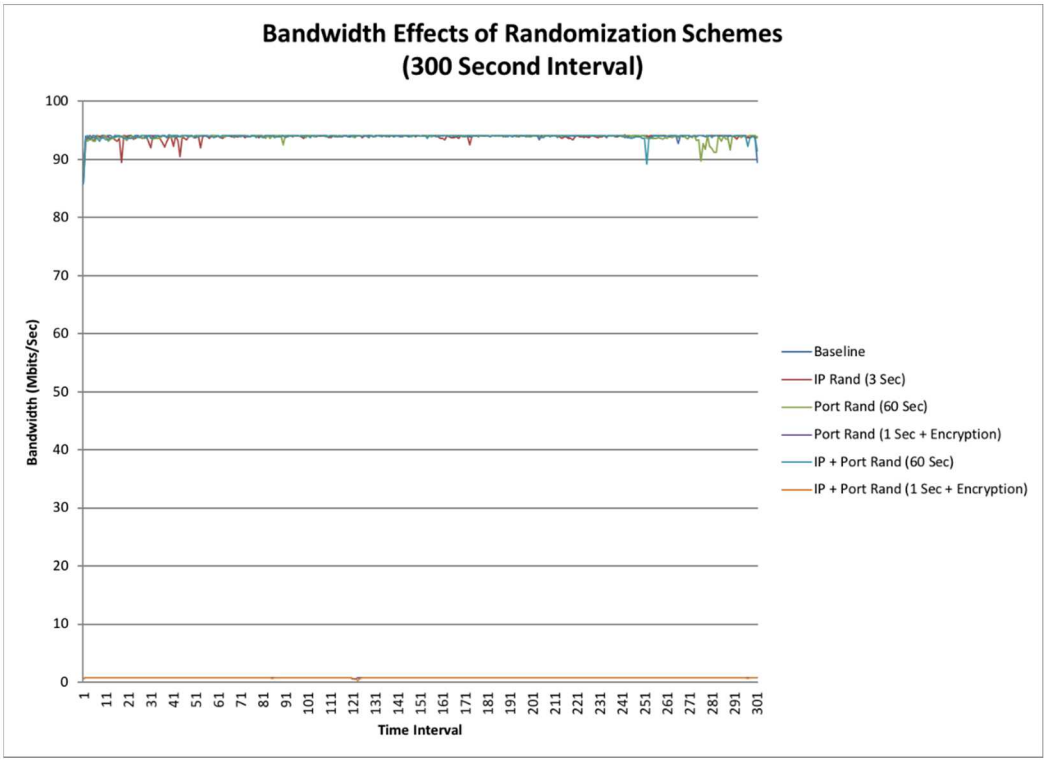

Figure 6. ***The measured bandwidth when enabling each randomization scheme independently and in combination of each other over one second increments in a 300 second interval of time.***

Figure 7 and Figure 8 show the results of the number of retransmits incurred when each of the randomization schemes were enabled independently and in combination with one another. When measuring the baseline configuration without any of the randomization schemes enabled, 124 retransmits were recorded, or ~0.005% of the total number of packets. When enabling the IP randomization scheme and the port randomization every 60 seconds scheme independently, there were fewer retransmits and a fewer percentage of overall retransmits. This may be a result of a more congested network during the time that the baseline measurements were recorded. It would be expected that the baseline would be similar or better than when enabling each of the randomization schemes. The remaining three schemes all produced a higher percentage of retransmits, although only two had a higher number of total retransmits. This is due to fewer packets being transmitted when randomizing port numbers every second. Although the percentages in all cases are low, it should be considered whether these percentages are acceptable to be applied within an operational setting. The results obtained were captured using the same iperf command that was specified for the bandwidth and throughput measurements, as shown in Figure 7 and Figure 8. The number of dropped packets were also measured as part of these tests. All of the schemes reported no dropped packets.

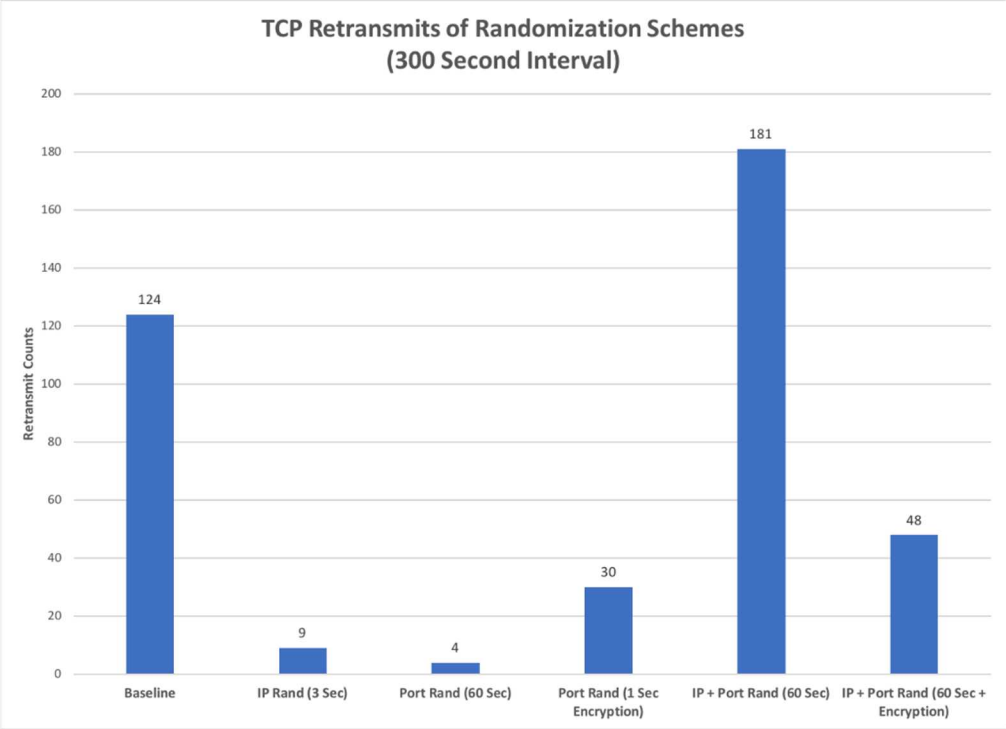

Figure 7. ***The total number of TCP retransmits recorded when each of the randomization schemes were enabled independently and in combination of each other over a 300 second interval of time.***

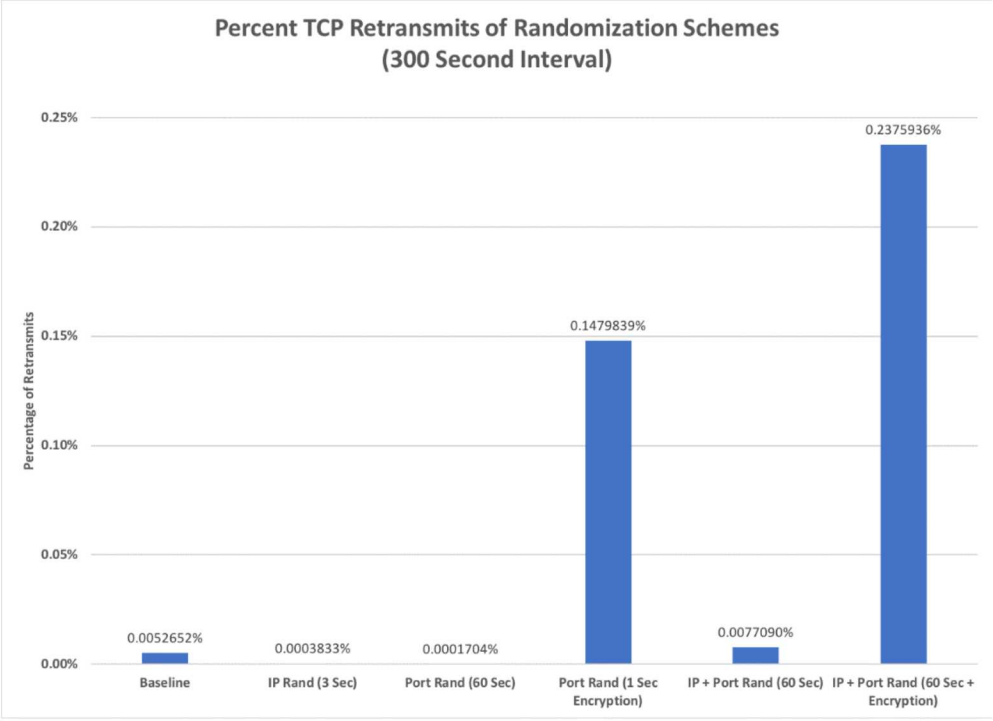

Figure 8. ***The percentage of TCP retransmits recorded when each of the randomization schemes were enabled independently and in combination of each other over a 300 second interval of time.***

5. CONCLUSIONS

The evaluated network-based MTD approaches have been shown to be effective within an ICS environment. We performed several experiments with a variety of configurations for each MTD technique. The techniques presented, although effective individually, are meant to be a piece of the larger computer security puzzle. These MTD techniques can be thought of as additional layers of defense to help protect a system from an adversary attempting to gain an understanding of a system in the early stages of an attack. Additional defenses can be deployed alongside the MTD techniques to create an even more secure system. Deploying an individual MTD technique or a suite of MTD techniques alongside other computer security protections will depend on the application. For example, the MTD techniques may provide a way to mitigate a "hitlist" type of attack [77], but the MTD techniques themselves do not provide the ability to detect the hitlist attack. Intrusion detection systems (IDSs), firewalls, security information and event management (SIEM) systems, and virus scanners, for example, should all be included as part of the overall security protection, as well. In this scenario, we used machine-learning algorithms to detect the hitlist attack and trigger the MTD schemes. The MTD strategies by themselves are not meant to be a comprehensive security solution that protects against all threats, but rather should be applied as an additional layer of defense in general.

6. ACKNOWLEDGEMENTS

This research presented in this chapter was funded by the Cybersecurity for Energy Delivery Systems (CEDS) Program within the U.S. Department of Energy/Office of Electricity Delivery and Energy Reliability (DOE/OE). This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

1. Stouffer, K, Falco, J., and Scarfone, K., "Guide to Industrial Control Systems (ICS) Security," NIST Special Publication 800-82, 2011.
2. Chandia, R., Gonzalez, J., Kilpatrick, T., Papa, M., and Shenoi, S. "Security Strategies for SCADA Networks," in *Critical Infrastructure Protection*, pp. 117–131, Springer, 2007.
3. Cardenas, A., Amin, S., Lin, Z., Huang, Y., Huang, C., and Sastry, S. "Attacks Against Process Control Systems: Risk Assessment, Detection, and Response," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pp. 355–366, ACM, 2011.
4. Huang, Y., Cardenas, A., Amin, S., Lin, Z., Tsai, H., and Sastry, S. "Understanding the Physical and Economic Consequences of Attacks on Control Systems," *International Journal of Critical Infrastructure Protection*, vol. 2, no. 3, pp. 73–83, 2009.
5. Miller, B., and Rowe, D. "A Survey SCADA of and Critical Infrastructure Incidents," in *Proceedings of the 1st Annual Conference on Research in Information Technology*, pp. 51–56, ACM, 2012.
6. Falliere, N., Murchu, L., and Chien, E. "W32. Stuxnet Dossier," White paper, Symantec Corp., Security Response, vol. 5, 2011.
7. Pourbeik, P., Kundur, P., and Taylor, C., "The Anatomy of a Power Grid Blackout," *IEEE Power and Energy Magazine*, vol. 4, no. 5, pp. 22–29, 2006.
8. Liang, G., Weller, S., Zhao, J., Luo, F., and Dong, Z. "The 2015 Ukraine Blackout: Implications for False Data Injection Attacks," *IEEE Transactions on Power Systems*, vol. 32, pp. 3317–3318, July 2017.
9. Sridhar, S., and Govindarasu, M. "Data Integrity Attacks and Their Impacts on SCADA Control System," in *IEEE PES General Meeting*, pp. 1–6, July 2010.
10. Farhangi, H. "The Path of the Smart Grid," *Power and Energy Magazine*, IEEE, vol. 8, no. 1, pp. 18–28, 2010.
11. Robles, R., Choi, M., Cho, E., Kim, S., Park, G., and Lee, J. "Common Threats and Vulnerabilities of Critical Infrastructures," *International Journal of Control and Automation*, vol. 1, no. 1, pp. 17–22, 2008.
12. Hauser, C., Bakken, D., and Bose, A. "A Failure to Communicate: Next Generation Communication Requirements, Technologies, and Architecture for the Electric Power Grid," *IEEE Power and Energy Magazine*, vol. 3, no. 2, pp. 47– 55, 2005.
13. Rajkumar, R., Lee, I., Sha, L., and Stankovic, J. "Cyber-Physical Systems: The Next Computing Revolution," in *Design Automation Conference*, pp. 731–736, June 2010.
14. Ericsson, G. "Cyber Security and Power System Communication Essential Parts of a Smart Grid Infrastructure," *IEEE Transactions on Power Delivery*, vol. 25, no. 3, pp. 1501–1507, 2010.
15. Hofmeyr, S., and Forrest, S. "Architecture for an Artificial Immune System," *Evolutionary Computation*, vol. 8, no. 4, pp. 443–473, 2000.
16. Al-Shaer, E., Duan, Q., Jafarian, J. "Random Host Mutation for Moving Target Defense," in *SecureComm*, pp. 310–327, Springer, 2012.
17. Antonatos, S., Akritidis, P., Markatos, E., and Anagnostakis, K. "Defending Against Hitlist Worms Using Network Address Space Randomization," *Computer Networks*, vol. 51, no. 12, pp. 3471–3490, 2007.
18. Farris, K., and Cybenko, G. "Quantification of moving target cyber defenses," in *SPIE Defense+ Security*, pp. 94560L–94560L, International Society for Optics and Photonics, 2015.
19. Sharon, A., Levy, R., Cohen, Y., Haiut A., Stroh, A., Raz, D. "Automatic Network Traffic Analysis," Oct. 24 2000. U.S. Patent 6,137,782.

20. Goldschlag, D., Reed, M., and Syverson, P. "Onion Routing for Anonymous and Private Internet Connections," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, 1999.
21. Shmatikov, V., and Wang, M. "Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses," *Computer Security–ESORICS 2006*, pp. 18–33, 2006.
22. Raymond, J. "Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems," in *Designing Privacy Enhancing Technologies*, pp. 10–29, Springer, 2001.
23. Dingledine, R., Mathewson, N., and Syverson, P. "Tor: The Second-Generation Onion Router," in *Usenix Security*, 2004.
24. "The Tor Project," 2014. Available online: https://metrics.torproject.org/torperf.html.
25. Chakravarty, S., Barbera, M., Portokalidis, G., Polychronakis, M., and Keromytis, A. "On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records," in *PAM*, pp. 247–257, Springer, 2014.
26. Tchabe G., and Xu, Y. "Anonymous Communications: A Survey on I2P," CDC Publication Theoretische Informatik-Kryptographie und Computeralgebra, 2014. Available online: https://www.cdc.informatik.tu-darmstadt.de.
27. Keromytis, A., Misra, V., and Rubenstein, D. "SOS: An Architecture For Mitigating DDoS Attacks," *IEEE Journal of Selected Areas in Communications*, vol. 22, no. 1, pp. 176–188, 2004.
28. Ahsan, K., and Kundur, D. "Practical Data Hiding in TCP/IP," in *Proc. Workshop on Multimedia Security at ACM Multimedia*, vol. 2, 2002.
29. Pimenidis, L., and Kolsch, T. "Transparent Anonymization of IP Based Network Traffic," In *Proceedings of 10th Nordic Workshop on Secure IT-Systems*, 2005.
30. Jajodia, S., Ghosh, A., Subrahmanian, V., Swarup, V., Wang, C., and Wang, X. (Eds.) *Moving Target Defense II – Application of Game Theory and Adversarial Modeling*, Springer, 2013.
31. Okhravi, H., Rabe, M., Mayberry, T., Leonard, W., Hobson, T., Bigelow, D., and Streilein, W. "Survey of Cyber Moving Target Techniques," Massachusetts Institute of Technology, Lexington Lincoln Laboratory Technical Report, 2013.
32. Salamat, B., Jackson, T., Wagner, G., Wimmer, C., and Franz, M. "Runtime Defense Against Code Injection Attacks Using Replicated Execution," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 4, pp. 588–601, 2011.
33. Holland, D., Lim, A., and Seltzer, M., "An Architecture a Day Keeps the Hacker Away," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 1, pp. 34–41, 2005.
34. Okhravi, H., Comella, A., Robinson, E., and Haines, J. "Creating a Cyber Moving Target for Critical Infrastructure Applications Using Platform Diversity," *International Journal of Critical Infrastructure Protection*, vol. 5, no. 1, pp. 30–39, 2012.
35. Cox, B., Evans, D., Filipi, A., Rowanhill, J., Hu, W., Davidson, J., Knight, J., Nguyen-Tuong, A., and Hiser, J. "N-Variant Systems: A Secretless Framework for Security Through Diversity," in *USENIX Security Symposium*, pp. 105–120, 2006.
36. McLaughlin, S., Podkuiko, D., Delozier, A., Miadzvezhanka, S., and McDaniel, P. "Embedded Firmware Diversity for Smart Electric Meters," in *HotSec'10 Proceedings of the 5th USENIX Conference on Hot Topics in Security*, 2010.
37. Kc, G., Keromytis, A., and Prevelakis, V. "Countering Code-Injection Attacks with Instruction-Set Randomization," in *Proceedings of the 10th ACM conference on Computer and Communications Security*, pp. 272–280, ACM, 2003.
38. Shacham, H., Page, M., Eu-Jin Goh, B., Modadugu, N., and Boneh, D. "On the Effectiveness of Address-Space Randomization," in *Proceedings of the 11th ACM conference on Computer and Communications Security*, pp. 298–307, ACM, 2004.

39. Sovarel, A., Evans, D., and Paul, N. "Where's the FEEB? The Effectiveness of Instruction Set Randomization," in *USENIX Security Symposium*, 2005.
40. Ganz J., and Peisert, S. "ASLR: How Robust is the Randomness?," in *Proceedings of the 2017 IEEE Secure Development Conference (SecDev)*, 2017.
41. Chavez, A., Stout, W., and Peisert, S. "Techniques for the Dynamic Randomization of Network Attributes." *Proceedings of the 49th Annual International Carnahan Conference on Security Technology*. 2015.
42. Forrest, S., Somayaji, A., and Ackley, D. "Building Diverse Computer Systems," in *The Sixth Workshop on Hot Topics in Operating Systems*, pp. 67–72, IEEE, 1997.
43. Cadar, C., Akritidis, P. Costa, M., Martin, J., and Castro, M. "Data Randomization," Technical Report TR-2008-120, Microsoft Research, 2008.
44. Boyd, S., and Keromytis, A. "SQLrand: Preventing SQL Injection Attacks," in *Applied Cryptography and Network Security*, pp. 292–302, Springer, 2004.
45. O'Donnell, A., and Sethu, H. "On Achieving Software Diversity for Improved Network Security using Distributed Coloring Algorithms," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pp. 121–131, ACM, 2004.
46. Zhang Q., and Reeves, D. "Metaaware: Identifying Metamorphic Malware," in *Twenty-Third Annual Computer Security Applications Conference, ACSAC 2007*, pp. 411–420, IEEE, 2007.
47. Rieback, M., Crispo, B., and Tanenbaum, A. "Is Your Cat Infected with a Computer Virus?," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, pp. 10, IEEE, 2006.
48. You I., and Yim, K. "Malware Obfuscation Techniques: A Brief Survey," in *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, pp. 297–300, IEEE, 2010.
49. Butts, J., and Sohi, G. "Dynamic Dead-Instruction Detection and Elimination," *ACM SIGOPS Operating Systems Review*, vol. 36, no. 5, pp. 199–210, 2002.
50. Sung, A., Xu, J., Chavez, P., and Mukkamala, S. "Static Analyzer of Vicious Executables (Save)," in *20th Annual Computer Security Applications Conference, 2004*. pp. 326–334, IEEE, 2004.
51. Salamat, B., Gal, A., and Franz, M. "Reverse Stack Execution in a Multi-Variant Execution Environment," in *Workshop on Compiler and Architectural Techniques for Application Reliability and Security*, pp. 1–7, 2008.
52. Min, B., Varadharajan, V., Tupakula, U., and Hitchens, M. "Antivirus Security: Naked During Updates," *Software: Practice and Experience*, vol. 44, no. 10, pp. 1201–1222, 2014.
53. Sutter, B., Anckaert, B., Geiregat, J. Chanet, D., and De Bosschere, K. "Instruction Set Limitation in Support of Software Diversity," in *Information Security and Cryptology, ICISC 2008*, pp. 152–165, Springer, 2009.
54. Kuck, D., Kuhn, R., Padua, D., Leasure, B., and Wolfe, M. "Dependence Graphs and Compiler Optimizations," in *Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 207–218, ACM, 1981.
55. North American Electricity Council (NERC) Critical Infrastructure Protection (CIP) Reliability Standards, 2009. Available online: http://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx.
56. Disterer, G. "ISO/IEC 27000, 27001 and 27002 for Information Security Management," *Journal of Information Security*, vol. 4, no. 02, p. 92, 2013.
57. National Institute of Standards and Technology (NIST) Cybersecurity Framework (CSF), 2014. Available online: https://www.nist.gov/cyberframework.

58. LaCommare K., and Eto, J. “Cost of Power Interruptions to Electricity Consumers in the United States (U.S.),” *Energy*, vol. 31, no. 12, pp. 1845–1855, 2006.
59. Andersson, G., Donalek, P., Farmer, R., Hatziargyriou, N., Kamwa, I., Kundur, P., Martins, N., Paserba, J., Pourbeik, P., Sanchez-Gasca, J., Shultz, R., Stankovic, J., Taylor, C., and Vittal, V. “Causes of the 2003 Major Grid Blackouts in North America and Europe, and Recommended Means to improve System Dynamic Performance,” *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1922–1928, 2005.
60. U.S.-Canada Power System Outage Task Force, “Final Report on the August 14, 2003, Blackout in the United States and Canada: Causes and Recommendations,” Merrimack Station AR-1165. Available online: https://www3.epa.gov/region1/npdes/merrimackstation/pdfs/ar/AR-1165.pdf.
61. Anderson, G., and Bell, M. “Lights Out: Impact of the August 2003 Power Outage on Mortality in New York, NY,” *Epidemiology* (Cambridge, Mass.), vol. 23, no. 2, p. 189, 2012.
62. IDA Modbus, “Modbus Application Protocol Specification v1. 1a,” North Grafton, Massachusetts (www. modbus. org/specs. php), 2004.
63. Clarke, G., Reynders, D., and Wright, E. Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems. Newnes, 2004.
64. Joachim, F. “PROFINET-Scalable Factory Communication for all Applications,” in *Proceedings of Factory Communication Systems, 2004*. pp. 33–38, IEEE, 2004.
65. Walters, R. “Cyber Attacks on U.S. Companies in 2014,” The Heritage Foundation, vol. 4289, pp. 1-5, 2014.
66. Zhu, B., Joseph, A., and Sastry, S. “A Taxonomy of Cyber Attacks on SCADA Systems,” in Internet of things (iThings/CPSCom), *4th International Conference on Cyber, Physical and Social Computing*, pp. 380–388, IEEE, 2011.
67. Miller, F., Vandome, A., and McBrewster, J. “Advanced Encryption Standard,” 2009.
68. Chodowiec, P. “Comparison of the Hardware Performance of the AES Candidates using Reconfigurable Hardware.” Ph.D. thesis, George Mason University, 2002.
69. Robertson, F., Carroll, J., Sanders, W., Yardley, T., Heine, E., Hadley, M., McKinnon, D., Motteler, B., Giri, J., Walker, W., and McCartha, E. “Secure Information Exchange Gateway for Electric Grid Operations,” Grid Protection Alliance Technical Report, Chattanooga, TN (United States), 2014.
70. Hurd, S., Stamp, J., and Chavez, A. “OPSAID Initial Design and Testing Report,” Department of Energy, 2007.
71. Smith, B., Stewart, J., Halbgewachs, R., and Chavez, A. “Cyber Security Interoperability: The Lemnos Project,” in *53rd ISA POWID Symposium*, vol. 483, pp. 50–59, 2010.
72. Halbgewachs, R., and Chavez, A. “OPSAID Improvements and Capabilities Report,” Sandia National Laboratories Technical Report, 2011.
73. Hughes, T. Networks of Power: Electrification in Western Society, 1880–1930. JHU Press, 1993.
74. Castello, P., Ferrari, P., Flammini, A., Muscas, A., and Rinaldi, S. “An IEC 61850-Compliant Distributed PMU for Electrical Substations," in *Applied Measurements for Power Systems (AMPS), 2012 IEEE International Workshop*, pp. 1–6, IEEE, 2012.
75. Milano F., and Anghel, M. “Impact of Time Delays on Power System Stability,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 4, pp. 889–900, 2012.
76. Mackiewicz, R. “Overview of IEC 61850 and Benefits,” in *Power Systems Conference and Exposition, 2006. PSCE’06.* 2006 IEEE PES, pp. 623-630, IEEE, 2006.
77. Staniford, S., Paxson, V., and Weaver, N. “How to Own the Internet in Your Spare Time,” in *USENIX Security Symposium*, vol. 2, pp. 14–15, 2002.