Dynamic Attack Surfaces in Nuclear Power Plants

Christopher C. Lamb and Lon A. Dawson
Sandia National Laboratories
1515 Eubank SE
Mail Stop 0933
Albuquerque, NM 87123
cclamb@sandia.gov; ladawso@sandia.gov

ABSTRACT

In this paper, we will summarize a group of architectural principles that inform the development of secure control system architectures, followed by a methodology that allows designers to understand the attack surface of components and subsystems in a way that supports the integration of these surfaces into a single attack surface. We will then show how this methodology can be used to analyze the control system attack surface from a variety of threats, including knowledgeable insiders. We close the paper with an overview of how this approach can be folded into a more rigorous mathematical analysis of the system to define the system's security posture.

Key Words: attack graphs, cyber security, control systems

1 INTRODUCTION

Over the past decade, cyber security has become much more important to digital systems, including control systems. While there has been a figurative flood of work with respect to securing enterprise computer systems, study of control system architectures is lagging. It becomes more and more important every day that we understand the security postures of these systems as they are increasingly under attack. With enterprise computer systems a breach may result in lost data, credit card information, or trade secrets. A breach in a safety critical system in a nuclear power plant can have much worse and longer-lived consequences.

The primary contribution of this paper is the flexible attack surface derivation methodology and the mathematical analysis of that surface to provide a cyber security model of the control system. Attack surface analysis is not new, but the way in which the surface is created and analyzed is currently inconsistent. This work will demonstrate a way to describe these surfaces that is repeatable, composable, and supports analysis thus enabling secure control system design.

We will first summarize relevant cyber security principles and their priority and application to nuclear power standards in Section 2. Then, in Section 3, we will begin to cover how define an attack surface, starting with a simple example to illustrate attack surface composition and definition. We will then define how attack surfaces can be aggregated more formally, leading into how specifically attack surfaces change based on the attacker's access and attack progression. Then, we will describe some of the properties of attack surfaces, including their non-homogenous Markovian nature. Following that, we will show how we can predict how the system's attack surface changes based on new environmental threats, and how engineers can use this model control system cyber security.

2 CYBER SECURITY PRINCIPLES RELEVANT TO NPP

Engineers have been researching cyber security principles for the past forty years. These principles have been published in a variety of forums, and most recently summarized in a group of technical reports[1,

2]. The identified principles have stabilized over this time period, and today, we have an understanding of what these principles are, and when they should be applied. These principles provide guidance for security engineers and visibility into potential areas for security improvement. While most of these principles apply to control systems, some are particularly important considering cyber security. In this section, we will point out those specific principles that need focus and where they are outlined in current standards documents.

2.1 Cyber security Priorities

Prior to outlining important principles, we need to define the cyber security priorities. In a cyber security analysis, common priorities are confidentiality, integrity, and availability. Authentication is frequently included as well. But for control systems, confidentiality is not nearly as important as integrity. After all, we can always determine the content of a system directive based on actuator response, but we need to make sure that the directive gets where it needs to go first. It is vital that control systems are available, and that components only respond to authorized directives.

Standard's require that NPP control systems include segmentation of function and strong isolation from external or enterprise networks [3, 4]. This makes NPP control systems relatively robust to external cyber attacks but insider attacks are still a concern. As a result, authentication (and by extension, authorization) is a key NPP control system priority.

From our perspective, key priorities for NPPs are integrity and availability, then authentication, followed by confidentiality. Control commands must be able to reach the actuator from the processing system, and sensor data must be able to reach the processing system from the sensor for the physical system to be controlled.

2.2 Key Principles

System **availability** is crucial for control systems. Systems are designed to be highly available, and standards codify approaches to availability and failure as well[3, 4]. Engineers have and continue to put much thought into the states devices can fail into as well as anticipating the failure response of nuclear power systems. Nuclear power systems work with hazardous materials and huge amounts of energy. If systems fail, it is vital that they fail into a safe state. Likewise, dependent systems should be able to handle component failure, and designs consider the implications of a component failure on a larger system.

Authentication is addressed in relevant standards, as is **defense in depth** and **deny by default** approaches[3, 4]. Authentication is vital to securely operating any system, and references to authentication requirements are found throughout the relevant standards. Exception processes for authentication exist as well, though potential mitigations for attacks on unauthenticated command infrastructure are not specifically addressed beyond data logging and physical security. Defense in depth approaches are likewise stressed in standards, though not overly prescriptively. Some specific controls, like intrusion detection systems and data diodes, are specifically called out however. Finally, deny by default strategies are usually the cornerstone of effective firewall ruleset design, and the standards documents correctly address this approach.

Nuclear power plants (NPPs) today **manage sensitive data** and maintain **separate data and control** networks via embedded data diodes and firewalls[3, 4]. These principles are currently recognized by specific technical controls in today's fleet. Sensitive data can be protected via **encryption** in NPPs as well[3, 4]. Standard guidance exists for establishing public key infrastructure, using encryption to protect network traffic, and the selection and use of various cryptographic modules. **Least privilege** and **separation of duty** are related controls that are also addressed in standards documents[3, 4]. The principle of least privilege restricts personnel and systems to the smallest set of capabilities needed to perform assigned duties or actions. Separation of duty is a common strategy used to allow for systemic checks and balances by restricting the responsibilities of users, engineering personnel, and collaborating systems. **Separation of privilege** is like separation of duty, but addresses providing checks and balances between different sets of

authority. This approach, separating privilege into distinct sets, is a common security strategy used in distributed computer systems. It is addressed obliquely within standardization documents, but not nearly as strongly as either of the previous two principles. Both **access controls** and information **flow controls** are also addressed in standards documents[3, 4]. The former from the perspective of controlling access to sensitive systems and components, and the later from ensuring information is appropriately protected in transit.

Compromise recording is a core part of the relevant cyber security standards as well[3, 4]. These standards describe how to implement defense in depth, the importance of intrusion detection, and how to guard information integrity. These standards guide engineers to configure logging and auditing systems to capture suspicious events and log them, acting on those events if appropriate. Compromise recording is key to detecting and responding to possible malicious activity. Furthermore, collected data can be analyzed offline to scan for resident persistent threats.

The category **Manage Your Assets** has importance in NPP cyber security standards and practices, most notably the **Validate Information** principle. Validate Information ensures that transmitted data is well formed and will not compromise receivers.

Keep It Simple as a category is always good guidance, but some systems are still complex even when they are as simple as possible. NPPs likely fall into this category[3-5]. **Least Common Mechanism** is an important principle to protect against common cause failure. The design basis for ensuring NPP reliability helps implement this principle.

2.3 Design Principles and Attack Surfaces

Design principles are important as they directly impact the exposed attack surface. In fact, many of the principles are designed to do just that – to shrink the surface visible to an attacker. For example, both **Least Common Mechanism** and **Separation of Privilege** were explicitly designed to shrink possible attack surfaces. Likewise, **Minimize Secrets**, **Reduce Sensitivity** and **Reduce Exposure** directly contribute to attack surface reduction as well. These principles when applied during system design or via security control application contribute to making systems more secure, often reducing the attack surface.

3 ATTACK SURFACE MANGEMENT

When designing a system, or refactoring a system to enhance security, we need a way to evaluate the security posture of that system both before and after the application of a technique or control. In this section, we will begin to outline such a method. Attack Surfaces are models of the vulnerabilities associated with a given system. They have been used extensively over the last decade to help highlight the attack vectors in systems[6-8]. They typically highlight possible routes of attacker ingress and ways attackers can exfiltrate stolen information (route of egress). These can further be separated into host based surfaces, network-centric surfaces, and at times, human or social surfaces.

In this analysis, we will limit the surfaces to network and host-based surfaces. We will not include social aspects in this analysis. Engineers could certainly extend this methodology to include human or social interactions, but at increased complexity that would not help outline our approach.

We will begin by outlining the attack surface of a USB connected thermocouple (believe it or not, it does have one, and it can be exploited). We will use this example to inform our intuition as we begin to assemble an attack surface. This first attack surface analysis is typical, in that we are looking at host and communication-based ingress and egress vectors. This thermocouple does not attach to a network, so it does not have a network-centric attack surface, but it does communicate with the host system over the attached USB cable. So typically, using non-hardened COTS components, we would use something like Omega's UTC USB thermocouple adapter[9] with, say, a J Type hollow tube thermocouple probe [9, 10].

Overall, this thermocouple and connector are very simple, and logically look something like Figure 1.

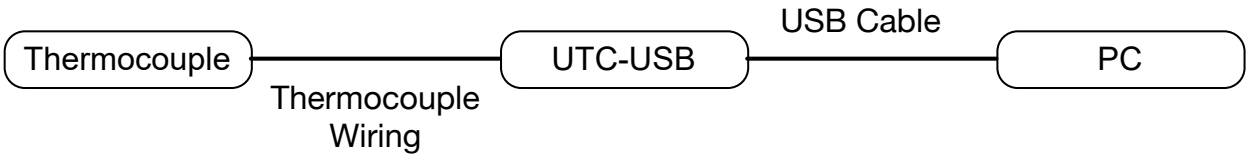

Figure 1: Notional thermocouple and adapter configuration.

The thermocouple itself does not have much of an attack surface. We have wires that route into the thermocouple, but they do not transmit any data, nor do they attach to a system that can be digitally exploited. Moving along the chain from left to right, we then come to the UTC-USB device. This is more complex, contains digital assets, as well as analysis software.

If you look at the spec sheet for the UTC-USB, you will notice that not only does it support a wide range of thermocouple types and connectors, it also "…includes Free Software for Display, Data Logging, and Chart Recording…" [9]. So, this connector not only connects to a PC via a USB cable, it will also attempt to install packaged software. Overall, we have two means of egress and ingress; the USB cable and the thermocouple wiring. And the connector itself uses one of these means of ingress to make software available to a PC host.

Moving on, we have the PC that collects data from the USB connected probe. The PC has a much larger attack surface than either of the previous components. It is not attached to a network, though it is a fully featured general purpose PC. In this example, we will assume that the PC is only used for data collection, debugging, and data transfer. It does not have any running network applications, nor is it ever attached to a network. Data is transferred to and from the PC manually using a USB pen drive. This gives us two routes for data ingress and egress – the USB cable attached to the converter, and the pen drive used to transfer data files from the PC.

Next, we will examine the host-based attack surface. The thermocouple really does not have a host-based attack surface. We have a simple hollow tube thermocouple probe attached to wiring. There's no digital host to examine.

The USB converter is much more interesting. This device has input wiring on one side and a USB connecter on the other. Internally, it has a processor as well as storage for the packaged data analysis software. In this example, we have not been able to fully disassemble the device, we will make some educated assumptions about the internals of the device based on other, similar devices.

For example, USB pen drives offer similar kinds of functionality, in that they provide storage services, data communication functions, and the like. Generally, a pen drive has a USB connector, acting as an intermediary between the flash memory in the pen drive and the computer the pen drive is attached to. It will also have a USB controller chip that manages writing/reading to/from onboard storage. Finally, the drive has a flash memory chip, an oscillator, various test points, and potentially expansion sockets for additional flash memory. Data flows into and out of the connector to the flash memory chip, managed by the controller.

We are most interested in the USB controller. The UTC-USB converter is storing software internally that is then installed on the attached host to facilitate data collection. There is furthermore some kind of computational power within the converter, although it is low-power and computationally slow.

Finally, we have a PC and the USB drive used to pull data from the PC. The PC has all the typical host targets any PC would have, but since it is not attached to a network, some of the more common host applications will be difficult to access. General purpose PCs have large, complex attack surfaces with dedicated tooling to help you understand how that surface evolves[11], so we will not go into detail about how to undertake this kind of analysis. The USB drive on the other hand is simple, and has the same basic attack surface as the thermocouple converter.

Table I. Summary of Attack Surfaces

Component	Host	Communication
Thermocouple	N/A	(*ingress/egress*) Attached Wire
Converter	USB Controller Packaged Software	(*ingress/egress*) Thermocouple Wire (*ingress/egress*) USB Cable
PC	Windows OS	(*ingress/egress*) USB Cable (*ingress/egress*) Pen Drive
Pen Drive	USB Controller	(*ingress/egress*) PC (*ingress/egress*) PC Receiving Data

Now we have some idea of our attack surface. Not all of it is exposed though – individual components do have individual surfaces, as does the system. The system attack surface is not necessarily the sum of the surfaces of all the parts however.

In this case, imagine that the PC and the converter are both physically secured, and the key to the room is controlled. Let's furthermore assume that the converter is heavily vetted and the contained software has already been installed and analyzed to ensure security.

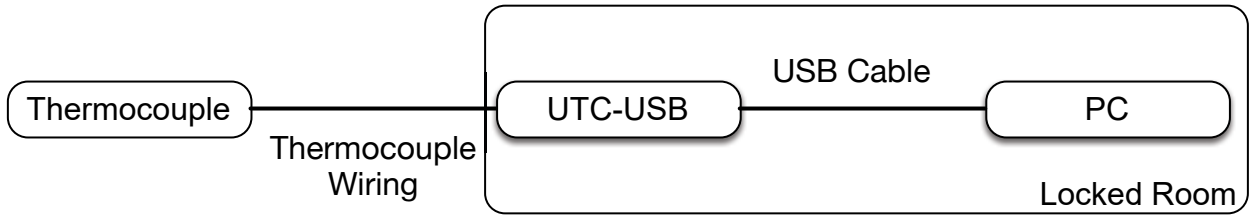

Figure 2: Physically secured components.

With these assumptions in place, we can treat the converter and the PC as a single component. This component also has, as we can see from Figure 2, two ways of communication ingress/egress; the thermocouple wiring and the USB drive used to pull data from the PC. This has essentially shrunk our attack surface by over 30%, simply by eliminating the converter and PC as well as the USB cable vector. Furthermore, the thermocouple wiring is a very low risk attack vector, so by combining the converter and the PC we have also removed some of the most egregious attack vectors in the system (i.e. USB cable access and converter access).

3.1 Evaluating Defenses Against Attackers

To evaluate the effectiveness of attack surface reduction efforts, we need some way to categorize attack vectors. Specifically, we need to be able to organize the vectors we have defined, and to track the vectors

themselves (even when they have been mitigated by a control or design strategy). We will examine how we can organize these vectors based on our new attack surface defined in Table II.

Table II. Summary of New Attack Surfaces

Component	Host	Communication
Thermocouple	N/A	(*ingress/egress*) Attached Wire
Converter/PC	USB Controller Packaged Software Windows OS	(*ingress/egress*) Thermocouple Wire (*ingress/egress*) Pen Drive
Pen Drive	USB Controller	(*ingress/egress*) PC (*ingress/egress*) PC Receiving Data

Table II lists a new aggregate attack surface for the system that is smaller than the combined surfaces of the components. With this in place, engineers need only deal with the new attack surface, in most cases. Unless, of course, the room is compromised.

If the secure room is compromised, or one of the other assumptions are rendered invalid, the attack surface will change again, reverting to the original, larger form. This kind of attack surface dynamism is rarely reflected in attack surface analysis today, which is considered more of a one-and-done function. Today, we build the attack surface at some point during system development, perhaps attempt to shrink it via deliberate design decisions or security control placement, and then we shelve it.

In the real world, where things change day-to-day and hour-to-hour, this just does not work.

3.2 Combining Attack Graphs

To effectively manage the attack surface of a system, we need to be able to manage the attack surface state in response to changing cyber conditions. And to do that, we need to understand what that attack surface is and how it transitions from state to state.

So, first, let's more formally describe what an attack surface is based on our previous example. Essentially, we can define an attack surface as a tuple containing four sets, as shown in Equation 1.

$$S = (I_h, E_h, I_c, E_c) \tag{1}$$

where I_h is the set of host ingress vectors, $E_h$is the set of host egress vectors, I_c is the set of communications or network ingress vectors, and E_c is the set of communications or network egress vectors.

We will also define an aggregation of attack surfaces as the union of the various vectors, such that if we have some number of components *C* the composite attack surface is the compilation of all possible attack vectors of the system components. Note that this definition does not consider design techniques, security controls, or perspectives into the system. In the previous example, the initial attack surface was the composition of all the component attack vectors, while the second attack surface used design controls and the point of view of a possible attacker to limit the attack surface, as shown in Equation 2.

$$S_t = S_1 \sqcup S_2 = (\{I_h^{S_1} \cup I_h^{S_2}\}, \{E_h^{S_1} \cup E_h^{S_2}\}, \{I_c^{S_1} \cup I_c^{S_2}\}, \{E_c^{S_1} \cup E_c^{S_2}\}) \tag{2}$$

We can extend this to define the combined attack surface of *C* components as the union of the contained sets of all *C* components. Note however that this is not invertible – if we have some attack surface defined by a set of subcomponents, it cannot be removed from the composite surface.

If we define the system to be the set C of system subcomponents, and the system attack surface as the composite surface of the attack surface defined in Equation 2, we can easily end up in a situation where the same attack vector is included in the attack surfaces of multiple subcomponents. In this case, we may remove a subcomponent with that vector, but because other subcomponents in C are affected by that vector, that vector still exists. Specifically, given $C = \{C_1, C_2, \dots, C_n\}$ and C_2, where $v \in I_h^{C_2}$, we don't have enough information to also know that $v \notin I_h^{C_x}, x \leq n$. Because of this, we cannot define an operation $\sqcap$ that easily inverts the $\sqcup$ operation. Essentially, all we can do is define $S_t \sqcap S_1 = S_2 \sqcup S_3 \sqcup \dots \sqcup S_n : S_t = S_1 \sqcup S_2 \sqcup \dots \sqcup S_n$.

To be able to apply set algebra to these constructs, we assume that the universal set of possible vectors does exist for each subset I and E, and is countable, though it is not knowable. By that we mean that it does exist, but at any given time the definition of that set may be hidden. This way we can also say that all the above sets have complements, and therefore set algebraic operations still apply.

3.3 Attacker Perspective

The previous operations took more of a horizontal, inclusive perspective on attack surface definitions. We also need to consider what an attack surface looks like from a particular point of view in a system, and how that attack surface changes as an attacker changes position in a system.

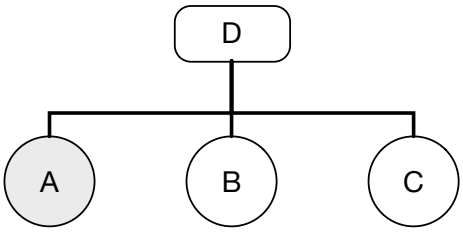

Figure 3: Simple notional networked system.

For example, in Figure 3, if we assume that host A is compromised, and that the system is self-contained with no external access, and that the switch at D prohibits communication between hosts A and C, then the initial system attack surface the attacker at host A can access is the composite surface of the switch D and the host B. As C is initially inaccessible, its attack surface is inconsequential.

Now, if switch D is compromised, then the attack surface at C is accessible and needs to be included in the current attack surface seen by the attacker. Note that while we are using a small LAN for simplicity, the approach still holds for host based analysis as well. This example has both topological restrictions (in that it is a standalone LAN) and controls (the switch blocks traffic between A and C) that limit the attack surface accessible to the attacker.

The attacker is essentially traversing a Markov chain while moving through a system. Formally, if we assume an initial state of a given system where the system is uncompromised, and that the states of the system are countable, we can say that the state X_{n+1} is dependent only on the state X_n, or that $P(X_{n+1}) = P(X_{n+1} | X_n, X_{n-1}, \dots, X_1) = P(X_{n+1} | X_n)$. After all, an attacker is only going to be able to extend an attack into a system from the previously compromised state of that system. If the system is uncompromised, the attacker needs to first gain a foothold. If the system is compromised, the attacker needs to pivot through the system using additional vulnerable elements.

While this model applies, the transition probabilities are not stationary. A given attacker may have a position in a system but not have the resources needed to take advantage of it until, one day, a new exploit is released that the attacker can use to further compromise that system.

This Markovian property of the set of states in a system representation allows us to evaluate the meaning and implications of a single state in isolation. This way, in order to understand the active attack surface of a system, we only need to keep the current state in mind. We do not need to maintain a record of previous states. This, in turn, makes formalizing attack graph derivation from a dynamic system more tractable.

3.4 Attacker's Perspective, Graph Management, and Dynamic Attack Graphs

We have described basic attributes of an attacker in a system. First, the process of an attacker transitioning a system is a non-homogenous Markov process with dynamic transition probabilities. We have also defined a way to aggregate and disaggregate attack graphs. Combining these capabilities, we can define a methodology to handle dynamic attack graph management.

First, let us define a system S. This system is defined by a set of components C related with a set of binary relations R over C. This allows us to define S as a graph where $S = (C, R)$. A relation $r = (c_m, c_n) \in R$ if c_n is visible from c_m, where visible means that an action from c_n will have some effect on c_m. If a component c_n is visible from another component c_m, and component c_m is compromised, then the attack surface of c_n is considered active. At any point in time, the active attack surface of S is the culmination of all active component attack surfaces modified by any applied security controls.

The system S can be characterized by a set of system states E. We have an initial state $e_o \in E$ that is the initial state of S. All states $e \in E$ are associated with an attack surface $u \in U$, including the initial state e_0. All states in E are related to other states in E via a set of binary relations B and a set of conditions A. This allows us to model the state space and associated transitions as a finite automaton, where E is the set of states; $\mathcal{P}(A)$, the powerset set of transition conditions, are the alphabet; $\delta : E \times \mathcal{P}(A) \longrightarrow E$ is the transition function; e_0 is the start state; and the set of final states is empty. The current state of this automaton is the current state of the system, and is associated with the active attack surface u of S.

Overall, this model allows us to determine the current attack surface of a system at a point in time, based on our best understanding of the state of the system. As an attack surface of a system will change in time based on current threats, we can estimate the current possible attack surface of the system from the perspective of attackers with a variety of roles, including malicious insiders.

Revisiting the system in Figure 2, we can define an initial attack surface for an external attacker as the Pen Drive or the thermocouple wire, as outlined in Table II. This is the initial state e_0. The system $S = \{thermocouple,\ Converter, PC,\ Pen\ Drive\}$ and $R = \{(Thermocouple, Converter), (Converter, PC), (PC, PenDrive\ )\}$. At e_0, the attack surface u associated with e_0 is described by Table II. We have some set of conditions A that model a threat information feed. The powerset, $\mathcal{P}(A)$, is the set of all combinations of threat information, and represents threat information that specifically informs the cyber security state of the system. The transition function δ takes, as input, the current state e_0 and the set of information from the threat feed, and outputs the new system state. The system is installed and we begin to monitor at time t_o. At some time t_n, we will assume that the threat feed informs us that the key to the locked room has been copied. With this information, the new system state output by the transition function is the state represented in Figure 1, as the room has been compromised, and the new attack surface is that described in Table I.

4 PREVIOUS WORK

Our proposed model for attack surface management differs from previous work in three areas. First, we are only interested in defining ingress and egress vectors; we make no attempt to evaluate the severity of a vector or the likelihood that the vector may be exposed. Rather, we are only interested in if a vector is exposed at all. Second, we do not model the system as an automaton, rather we model the system and its evolution as a Markov process with groups of related states. We model the process states and transitions as an automaton. We do define a system, but we are only interested in the structure of the system for our analysis. Finally, we are interested in evaluating how an attack surfaces changes based on external threat state rather than evaluating the likelihood of an attack or the risk of specific vectors.

Manadhata et al. define an attack surface in a similar way – which is not surprising as this is an industry standard definition today. They have been focused on defining the sizes of attack surfaces and the risks thereof. We use a more general definition of an attack surface in this work, including data items and communication channels as routes of ingress and egress. Our work diverges from there, as Manadhata et al. focus on developing metrics for attack surface definition based on damage potential and effort while we are concerned with evaluating the current attack surface of a system based on known threats and attack surface dynamics [12-16].

Howard et al. uses a state machine to define attacks through an attack surface. This analysis is like ours, but we use a more formal automaton and have a different focus. Howard focuses on vulnerabilities associated with an attack surface to derive a notion of attack surface size. With this size metric in hand, he then derives a notion of "attackability" of a system, which can then be used to make a system more secure by making the system less attackable. Manadhata and Howard use the same definition of an attack surface. We are more concerned with attack surface dynamics than attack surface evaluation [17].

Other authors have focused on more static qualitative definitions of attack surfaces and less automatable ways of attack surfaces management. Overall, the primary approach is defining the attack surface once, during system development, and using that surface definition to enhance overall security via surface reduction[7, 8].

5 CONCLUSIONS AND FUTURE WORK

Attack surface management has been a static exercise intended to highlight possible attack vectors in a system, and once defined, give developers a clear roadmap toward a more secure system. We extended this approach, showing how you can use the same basic guidelines in a more rigorous framework to dynamically evaluate the true attack surface of given system based on current system threats tied to potential attacker positioning in a system.

With this more formal description of an approach to dynamic system attack surface management, our next steps will focus on automating this analysis. Specifically, we intend to use this approach on a deployed system to help us proactively manage potential security vulnerabilities based on current threat feeds.

6 REFERENCES

1. Lamb, C. and J. Hatcher, *Attributes of Secureable Architectures*. 2015, Sandia National Laboratories.
2. Lamb, C., *A Survey of Secure Architectural Principles*. 2015, Sandia National Laboratories.
3. *Cyber Security Plan for Nuclear Power Reactors*. 2009, Nuclear Energy Institute.
4. *Cyber Security Programs for Nuclear Facilities*. 2010, Nuclear Regulatory Commission.

5. Protection of Digital Computer and Communication Systems and Networks. 2009, Nuclear Regulatory Commission.
6. OWASP. *Attack Surface Analysis Cheat Sheet*. 2015 07/18/2015 [cited 2017; Available from: https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet.
7. Olzak, T., Introduction to Enterprise Security: A Practitioner's Guide. 2012: INFOSEC Institute.
8. Northcutt, S. *The Attack Surface Problem*. 2017 [cited 2017; Available from: http://www.sans.edu/cyber-research/security-laboratory/article/did-attack-surface.
9. Omega. *Universal Thermocouple Connector Direct USB to PC Connection*. 2017 [cited 2017; Available from: http://www.omega.com/pptst/UTC-USB.html.
10. Omega. *Hollow Tube Thermocouple Probe*. 2017 [cited 2017; Available from: http://www.omega.com/pptst/HTTC36.html.
11. Microsoft. *Improving Security Using Attack Surface Analyzer*. 2017 [cited 2017 February 17]; Available from: https://technet.microsoft.com/en-us/security/gg749821.aspx.
12. Manadhata, P.K. and J.M. Wing, *A formal model for a system's attack surface*, in *Moving Target Defense*. 2011, Springer. p. 1–28.
13. Manadhata, P. and J.M. Wing, *Measuring a systems attack surface*. 2004, DTIC Document.
14. Manadhata, P.K., D.K. Kaynar, and J.M. Wing, *A formal model for a systems attack surface*. 2007, DTIC Document.
15. Manadhata, P.K. and J.M. Wing, *An attack surface metric*. IEEE Transactions on Software Engineering, 2011. **37**(3): p. 371–386.
16. Manadhata, P.K., et al., *An approach to measuring a systems attack surface*. 2007, DTIC Document.
17. Howard, M., J. Pincus, and J.M. Wing, *Measuring relative attack surfaces*, in *Computer Security in the 21st Century*. 2005, Springer. p. 109–137.