Security of Cyber-Physical Systems A Survey and Generalized Algorithm for Intrusion Detection and Determining Security Robustness of Cyber Physical Systems using Logical Truth Tables

Curtis G. Northcutt School of Engineering, Vanderbilt University

The recent proliferation of embedded cyber components in modern physical systems [1] has generated a variety of new security risks which threaten not only cyberspace, but our physical environment as well. Whereas earlier security threats resided primarily in cyberspace, the increasing marriage of digital technology with mechanical systems in cyber-physical systems (CPS), suggests the need for more advanced generalized CPS security measures. To address this problem, in this paper we consider the first step toward an improved security model: detecting the security attack. Using logical truth tables, we have developed a generalized algorithm for intrusion detection in CPS for systems which can be defined over discrete set of valued states. Additionally, a robustness algorithm is given which determines the level of security of a discrete-valued CPS against varying combinations of multiple signal alterations. These algorithms, when coupled with encryption keys which disallow multiple signal alteration, provide for a generalized security methodology for both cyber-security and cyber-physical systems.

Introduction

The recent proliferation of embedded cyber components in modern physical systems [1] has generated a variety of new security risks which threaten not only cyberspace, but our physical environment as well. Whereas earlier security threats resided primarily in cyberspace, the increasing marriage of digital technology with mechanical systems in cyber-physical systems (CPS), suggests the need for more advanced generalized CPS security measures. With the exponential increase [2] in CPS, many necessary security measures are forsaken as new areas are explored. For example, most modern day vehicles support OnStar services. Malicious signals can be sent via the OnStar Telecommunications network to easily remotely control your vehicle while you are driving as demonstrated by Koscher et al. (2010). [3]

Before considering these security issues, we must provide our definition of a cyber-physical system. A CPS, also referred to as an embedded system, is a conglomerate system composed of both computational and physical elements. Most CPS interact with their environment through sensors and actuators, forming a feedback loop which alters physical world. Sensors are used to input information about the environment; the CPS performs computations on this information and uses the results of these computations to control its actuators. These actuators in turn alter the environment, which changes the information input by the sensors and the cycle repeats. Examples of such CPS include SCADA industrial control systems, distributed temperature control systems, AMS (distributed control of home meters such as your electric meter), and traffic light infrastructures.

In defining CPS, it can be instrumental to compare cybersecurity with CPS security. Consider a localized cybersecurity attack where a hacker alters a person's bank account versus a localized CPS attack where a hacker accelerates a person's vehicle. In the former, the system can be restored, no physical harm is done, and no life is threatened. In the latter, the damage to the vehicle cannot be "deleted" in the way that electronic data can and is likely that the driver would be severely injured, possibly resulting in loss of life. We can also consider examples of larger scale. Consider a cybersecurity attack of altering the stock market versus a CPS attack of overheating a nuclear power plant. In the former, economic instability may ensue temporarily, but the market can be restored and there is no risk of human safety. In the latter, the power plant cannot be restored without building a new one and a power plant explosion could result in millions of deaths.

These make clear that greater important of CPS security over cybersecurity when considering the risks to human life. Furthermore, we can see cybersecurity as a subset of CPS security, since the cyber components of CPS require all cybersecurity protocols in addition to other security protocols introduced by the physical components and the interactions therein. This subset relationship suggests that CPS security methodologies also work for cybersecurity, but not vice versa, validating the greater need for increased efforts to understand CPS security as opposed to cybersecurity.

To address this problem, in this paper we consider the first step toward an improved security model: detecting the security attack. Using logical truth tables, we have developed a generalized algorithm for intrusion detection in CPS for systems which can be defined over discrete set of valued states. Our algorithm is composed of two parts: (1) *The intrusion detection algorithm* and (2) *The Robustness Algorithm.* The novelty is in the latter. The first algorithm either detects an intrusion immediately or returns a set of possible compromised signals. The second algorithm then determines the number of signals that can be compromised such that intrusion detection is guaranteed. This is the first algorithm which *guarantees* intrusion detection for systems which can be modelled as a discrete set of inputs and outputs.

Survey of Cyber-Physical Systems

A survey of topics, examples, and methodologies within CPS security is defined in this section.

Types of Security Attacks

There are two types of security attacks: passive and active [4] which are categorized in Table 1.

SCADA Systems

SCADA (Supervisory Control and Data Acquisition) Systems are industrial control systems composed of actuators and sensors that are centrally controlled by a mainframe. This mainframe uses the input from the sensors to adjust the output of the actuators in a feedback loop. In "*Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems*," Cardenas et. al. propose a taxonomy composed of the security properties of the sensor network, the threat model, and the security design space to understand the holistic nature of sensor network security for practical deployment. [5] They further described the goals of a sensor network and a guide to challenges that need to be addressed in order to prioritize our defenses against threats to application layer goals by providing a ranking of threats and security mechanisms similar to those described in the previous section.

Given the many known attacks on sensor networks, Cardenas et. al. emphasized that past research has been primarily devoted to countermeasures to prevent individual attacks. However, to ensure full generalized security against any attack, one would have to implement all of the proposed countermeasures into a single system, resulting in infeasible overhead. The problem addressed here is to define a protocol for systems to be secure against all general attacks (as opposed to individual countermeasures for each possible attack).

They contributed to this problem by defining the high-level security goals of a sensor network and identifying different ways that sensor measurements are

Table 1: Categories and Types of Security attacks

CATEGORY	TYPE	DESCRIPTION
Passive	Release of message contents	The unauthorized viewing of message contents.
	Traffic analysis (packet sniffing)	The observation of patterns of packets (connection and content data) sent over a network to obtain information about message content and user behavior.
Active	Masquerade Attack	A signal sent out which appears to be from a valid sender but is not.
	Replay Attack	Previous messages / signals are recorded and then sent at a later time, appearing to be the sender at the current time.
	Modification of Messages	Signals / Messages are intercepted, altered, and then sent to their original destination.
	Denial of Service	A distribution of compromised clients overload a server resulting in denial of service when the server is used by a valid client.

reported back to the base station since sensor measurements can compromise confidentiality of the network even when we use standard encryption algorithms.

Smart-Phone Systems

Smart-phones are unique in CPS as they provide a bridge between the telecommunications network and the internet via wireless and 3G/4G services. This bridge introduces a new type of security threat to both the telecom networks and internet networks. Malicious code written in multiple languages can manifest in the telecommunications network, then manifest within the mobile operating system, eventually transferring to the internet via the Wi-Fi or data connection and vice versa. This interconnection of multiple networks increases vulnerabilities and requires new considerations in how we secure network connections and smart-phones.

This problem is exacerbated by increasing computing power of smart-phones as well as their enormous prevalence in modern society. However, societal disinterest in the formalities of computers leads to minimal understanding of the similarities between computers and smart-phones, resulting in less than 1/5 of smart-phones having an installed security system. [6] The introduction of Near Field Communications (NFC), a new standard in mobile communication that uses radio signals to allow data transfer by touching two phones together in close proximity, has led to considerable close-range security risks. One such attack uses a NFC chip, that when passed near to a smart-phone with NFC enabled causes the phone to open a webpage which installs an unidentifiable remote access program. Another NFC hack is uses NFC to read data from a subway fare card and then transmit free rides to the persons' card. The increasing use of smart-phones and their interactions within a variety of networks including the internet, telecommunications, Bluetooth, Wi-Fi, and NFC confirm the considerable security risks of smart-phone CPS.

Stuxnet

Stuxnet, a virus which attacked the uranium treatment programmable logic controllers (PLCs) in an Iranian Nuclear Power Plant, is one of the most expensive cyber-physical systems attacks against a SCADA system. Researchers at Symantec, Falliere et. al. (2007), addressed the following questions [7]:

- How did Stuxnet infiltrate the Iranian SCADA system?
- How was Stuxnet designed?
- What effect did Stuxnet have on the Windows computers that it infected?
- What aspects of the SCADA system were affected by the virus?

The creation of Stuxnet resulted from a well-organized group of intelligent, well-trained programmers with insider information and stolen digitally-signed drivers (i.e. verified as legitimate drivers from a well-known corporation). Though the original intent of the attack is unknown, speculators have suggested that the attack was actually a ruse to distract from a greater cyber warfare effort. The Iranian Nuclear Plant used Siemen's machines running Windows Operating System. Stuxnet was introduced to these machines via a USB drive that was brought into the plant by an employee and inserted into a machine, as the SCADA machines were only connected via a local network. The virus is written in multiple languages and is inert unless on a particular SCADA machine with network access to the PLCs of the actuators running at a bounded frequency interval. Once the virus was loaded onto the computer, this computer infected the mainframe, which in turn infected the PLCs of the uranium treatment actuators, causing the treatment to fail, heavily damaging the nuclear plant.

Stuxnet then spread globally when a plant employee used their work computer on the internet in their home. The virus uploaded to servers worldwide affecting over 50% of Iranian computers. However, as the virus is specialized to affect the PLC controllers handling the uranium treatment, it had little effect on personal computers and was relatively harmless. The malware has both user-mode and kernel-mode rootkit capability under Windows, and its device drivers have been digitally signed with the private keys of two certificates that were stolen from separate companies, JMicron and Realtek, allowing it to go undetected by the operating system.

The global scope of Stuxnet requires us to rethink how we view CPS security. First, the local network isolation of the Iranian Power Plant ensured a high level of security, based on <u>cybersecurity</u> paradigms of that time. However, the initial infection was caused by <u>physical</u> means (USB insertion of virus). This demonstrates the need for CPS security paradigms for SCADA systems as cybersecurity alone is not enough. A further consideration is the multi-language nature of the virus which allowed it to spread from a USB to an OS to a local SCADA network, to individual PLC controllers, to the OS of a personal computer, and to the internet.

The virus was even able to emulate device drivers on the hardware interface level. This versatility resembles the issues discussed with smart-phone security systems. Thus, the introduction of physical components in traditional cyberspaces results in the need to consider multi-lingual virus possibilities in cyber-physical security protocols. The unique nature of Stuxnet allows it to be inert on personal computers. This leads to a variety of ethical considerations. Should we consider an inert virus a threat? How do we classify a virus that is not currently causing harm, but has latent potential to destroy other systems? These questions bring up ethical questions regarding the classification and harmfulness of viruses and suggest biological equivalences between the Stuxnet virus and diseases like Chlamydia, which are inert in males but can cause infertility when spread to females.

Stuxnet makes clear the impact of CPS security versus cybersecurity. A cyber-attack may alter an online infrastructure, but with backups and updates, these can be digitally restored. However, a cyber-physical attack can destroy a nuclear plant and millions of lives by the radiation caused from a nuclear meltdown, explaining the greater need for CPS security protocols.

Modern Automotive Hacking

With the introduction of telematics interfaces and On-Board Diagnostics ports, modern automobiles are connected to networks resulting in a variety of potential CPS security risks. As many of the newer vehicular electronic devices are relatively new, there are a considerable number of life-threatening vulnerabilities in the cyber infrastructure of modern vehicles, as demonstrated by Koscher et. al. (2010). There are two primary methods of remotely attaining unauthorized control of a vehicle:

1. In the United States, the federally-mandated On-Board Diagnostics (OBD-II) port, under the dash in virtually all modern vehicles, provides direct and standard access to internal automotive networks.
2. Telematic interfaces (like OnStar) bridge all important busses between Electronic Control Units to allow stopping a vehicle, unlocking doors, etc. These interfaces connect all parts of the vehicle to a cellular network.

Koscher's group focused on the security issues arising within a vehicle once the vehicle has already been infiltrated. They demonstrated adversarial control of a wide range of automotive functions and were able to completely ignore driver input—including disabling the brakes, selectively braking individual wheels on demand, stopping the engine, etc. – remarking that the "increased degree of computerized control brings with it a corresponding array of potential threats [3]."

Their work was unique to the field as previous research only described automotive security abstractly, whereas they described practical security issues on the road in practice, analyzed car digital components and internal networks, and evaluated security properties of each component and underlying network, demonstrating many security issues. By controlling the vehicle via the CAN bus (used for autonomous communication between microcontrollers within a vehicle), they were able to identify the following problems in the CAN bus network:

1. All packets broadcast to all nodes in the network, allowing easy manipulation of any connected component of the vehicle once access is obtained.
2. This distributed nature of the network was very fragile to Distributed Denial of Service attacks as well as looping packets to delay connections.
3. No authentication is needed to use the network, allowing easy infiltration.

Furthermore, Koscher et. al. found that many electronic component units (ECUs) in the vehicle did not follow standard required protocols, which allowed disabling communication in the network while the vehicle was moving. Using an understanding of the telematic interface, On-Board Diagnostics Port, CAN bus, and ECU vulnerabilities, they used two primary methodologies to alter the state of the vehicle:

1. The DeviceControl unit used by manufactures to test all ECUs could be used to send signal controls to any vehicle component.
2. Reverse engineering of packets sent on the CAN busses allowed masquerade attacks on any vehicle component.

Their work provided empirical evidence of the inadequacy of security protocols available in modern vehicles due to the lack of a cohesive set of components among all vehicles for monetary business reasons, the need for mechanics to have control for vehicular maintenance, and the need of DeviceControl by manufacturers for testing.

The Need For Artificial Intelligence in CPS Security

With the advent of more sophisticated malware and attacks, security based on defending against singular repeat attacks is inadequate. Artificially intelligent tools are needed to adapt to the variations in the malware. This current paradigm for security defense is purely reactive (for each individual attack, specify an individual counterattack), instead of an adaptive solution (an intelligent security agent that adapts to any type of attack using probability distributions.) Thus for attack A, the current paradigm suggests a counterattack A', however, when a new attack B is created, a reactive model fails to secure the system until a counterattack B' is conceived, thus suggesting the need for the adaptive model which maintains a system security specification regardless of new types of attacks. [8] Furthermore, most security solutions enforce results by inhibiting functionality or induced security software overhead reducing performance. However, there are considerable challenges that must be addressed when using sophisticated AI in CPS security. [9]

Morel et. al. (2011) used AI probability techniques to suggest a security detection that reduces the number of false positives, i.e. 99 false positives and 1 malware means 99% of the work done to correct things is wasted. Reducing false positives can be done using two primary techniques: [8]

1. Bayesian updating
   a. This method uses Bayes formula in Statistics to create a new variable that is updated with each incremental update of Bayes formula. By rearranging the equation in terms of false positives, we see that over time the probability of an attack occurring approaches one, eliminating the profound effect of false positives.
2. Factor improvement with networked computers using von Neumann's technique
   a. By increasing the number of computers, we can decrease the effect of false positives and increase the probability that an attack actually occurred. Consider logical 3-gates (binary logical gates with 3 input signals) where a false positive occurs if at least 2 of the gates submit false information. For low false positive values for each gate, the probability of a false positive is relative low, *p(fp) low*, but as input value increases, *p(fp) approaches 1.* By substituting this *p(fp)* into their original equation, it is possible to assume 9 computers instead of 3, which resulted in a decrease in *p(fp)*. For 81 computers, *p(fp)* is negligible.

Morel's work via these techniques provides a mechanism to ensure low false negatives <u>and</u> low false positives. However, two assumptions are made in the calculations – he assumes independence of probability of each computer and each gate and assumes all probabilities are equal, neither of which is true in practice. Regardless, this work demonstrates how artificial intelligence methodologies can be used to greatly reduced security system overhead, validating the usefulness and need for AI in CPS security.

Other Uses of AI in Security

AI techniques including game theoretic search, propositional logic, and Bayesian Networks are useful in modeling CPS security models using an adaptive approach. These techniques are described below, segmented in their respective categories.

1. Game Search using MiniMax with Alpha Beta Pruning [10]
   a. A minimax game tree in computer science is a decision tree where a protagonist tries to maximize their choices at every other level in the tree and an antagonist tries to minimize their choices in the levels of the tree between the protagonist's levels. Using AI, we can represent a security model using a minimax game tree. At each node in the tree, we have a value which represents the damage to the system. The security agent tries to minimize damage and the hacker tries to maximize damage. If at level *n*, it is the hacker's turn, then they will maximize choices among all nodes in the tree. Then at level *n-1*, the security agent will minimize choices among all nodes in the tree, and so on. As this process can be computationally intensive, Alpha Beta (AB) pruning is used to (sometimes greatly) increase the time efficiency of minimax, although in the worst case, AB pruning can provide no effect. AB pruning works by pruning nodes whenever no improvement can possibly be made down that path. The average reduction in time required to run this algorithm when using AB pruning with a limited depth bound suggests the feasibility of this game theoretic approach to a security model.
   b. This approach would be very effective to prevent system changes, but in its current description, would provide no protection against benign attacks (attacks which do not alter the state of the machine) such as attacks where the goal is to obtain private information. Also, the efficacy of this system on spoofing (mas-

querading with false signals to attain access to secure data) is debatable as the current description provides no means of determining the origins of data.

2. Propositional Logic and Symbolic Logic
    a. Logic systems are common in artificially intelligent agents. Given a set of axiomatic conditions, we can build up a set of theorems to search for a proof or contradiction. In security, we could define a set of axioms relating to necessary protocols every system must follow {private files are not accessed by unauthorized users, i.e. □ *pf, pf* ^ *au*, packets are authenticated, etc.} To gain access to a system, alter the system, or view private files to system, a proposition must be produced which is added to the set of axioms. If at any point, the knowledge base of axioms is not sound and a contradiction is found, we have detected an intrusion.
3. Bayesian Networks
    a. Consider the creation of a Bayesian Network for detecting a probable security attack. The joint probability distribution over the Bayesian Network describes the likelihood of an attack. Each node represents the probability of a specific security event, e.g. the probability that the admin account is compromised.
    b. By updating the singular probabilities in real-time, the joint distribution based on conditional probabilities could be used to determine the likelihood of an attack. Empirical usage would provide thresholds for these joint probabilities.

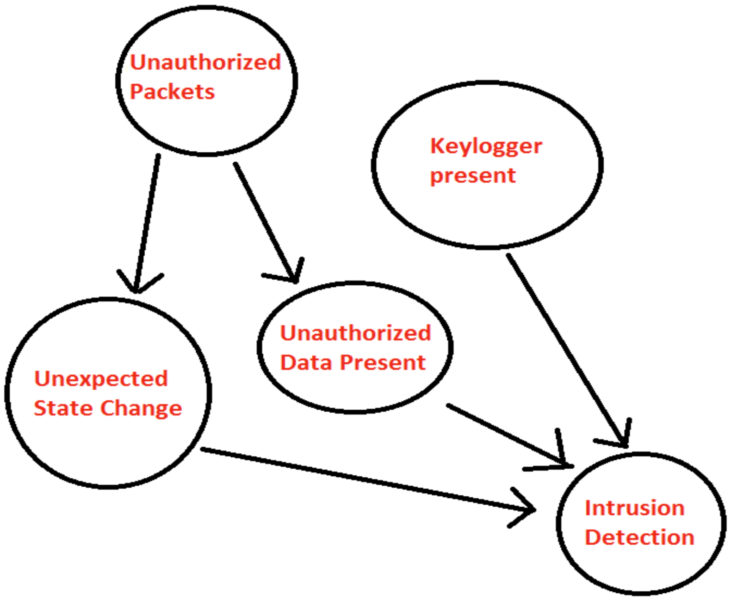

Figure 1: Sample Bayesian Network used for CPS security.

Generalized Methodology

Before considering the following algorithm, we must first define our terminology. Define *robustness* to be the cardinality of the *robust zone* set. The *robust zone* is a subset of the set of natural numbers, defined to contain an integer *i* iff the system guarantees an intrusion detection if *i* signals are compromised, where *intrusions* are defined as attacks which compromise signals. So, if for a given CPS, the *robust zone = [0, 1, 3]* then this states if 0, 1, or 3 of the signals in our system are compromised simultaneously, then we guarantee that we will detect the invalid state of the system, i.e. detect an *intrusion.* The greater the value of *robustness*, the greater the cardinality of the *robust zone*, and the more *robust* the security of the CPS.

We provide a generalized algorithm which yields two important security features:

1. Security Robustness of the system
    a. If no attack is detected, determine the robustness, *r*, of the system, i.e. given a system with *n* total inputs and outputs, we can determine whether our system can detect an attack if 1 signal is compromised, 2 signals are compromised,…, *n* signals are compromised, thus determining the strength of our system security. The robustness value determined describes the cardinality of the robust zone set.
2. Guarantee of intrusion detection within robust zone.
    a. If the system is in an invalid state, we immediately detect an intrusion.
    b. If the system is in a valid state, then we are guaranteed that we will detect an intrusion as long as the number of signals compromised is in the robust zone, as determined by the robustness value.

Construct a table (as shown by Table 1 in the next section) which contains an enumeration of all possible value combinations of all input and output signals and a binary signal labeled 'valid' for each row of the table which represents if the combination of input and output signals is a valid state given the constraints of the system. Define a *signal vector* to be a row in this table, not including the valid bit. Define *current signal vector* to be the current signal values of the system in real time. Once this table is constructed, we can use it to determine if our system has been compromised. In real time, consider the current system input values. If the *current signal vector* has a valid bit of 0, then the system is currently compromised. To detect which signal is compromised:

Intrusion Detection Algorithm (trivial):

```
    Create a set S of sets.
    For every combination of signals and values in the compromised signal vector:
        Toggle the current combination of signal values
        If the resultant signal vector is valid, add the set of 'toggled signals only' to S
```

This will result in a set S identifying all possible sets of compromised signals. If the current signal vector is valid, then we determine the robustness of our system.

Robustness Algorithm:

```
    robustArray = [true, true,…, true] of length |totalSignalsInSystem| + 1
    for num= 0 up to totalSignalsInSystem
        For every combination of 'num' number of signals/ values in current signal vector
            Toggle the current combination of signal values
            If toggled signal vector has valid bit == 1
                then set robustArray[num] = false
                break innermost loop

    Robust_Zone = empty set
    For i = 0 up to numSignals in System
        If robustArray[i]is true
            Robust_Zone.add_member(i)
```

The values in the Robust_Zone set is exactly the Robust zone as previously defined and thus provide the robustness of our system.

***Intuition of the Robustness Algorithm**:*

We defined which system states were valid in our table, thus if we are not in one of those states, we know automatically that system is compromised. If we are in a valid state, then we want to answer the following question: "Is it possible that an attacker altered the values of my signals to put me into another valid state such that now the output of the CPS does not match the inputs of the system, but the state still 'appears' to be valid?" The *robustness algorithm* provides an answer to this question by yielding a set, the robust zone, which contains all numbers of compromised signals such that no matter what the attacker changes, they cannot possibly change the current signal vector to be a signal vector which is a valid state (thus they cannot hide their alterations in the system and we guarantee that we can detect them). The *robustness algorithm* enumerates through all possible combinations to determine these values, so that we can guarantee intrusion detection for the values in the robust zone set.

Train Gate-Controller Example

Consider the CPS train gate-controller example as defined by Koutsoukos et. al. (2003). [11] In this simplified example, we have a gate that prevents cars from crossing a circular railroad by being raised or lowered. A train travels around the loop and we must control the gate based on the location of the train (see Figure 2).The location of the train is determined by two sensors, sensor1 and sensor2. Sensor1 is placed at location y = 5 and sensor2 is placed at location y = 15. When the train passes either sensor, both sensors toggle. Thus, the sensor signals cannot both be zero or one, otherwise this would imply the train is both before and after a sensor due to the circular track. In total, the system has

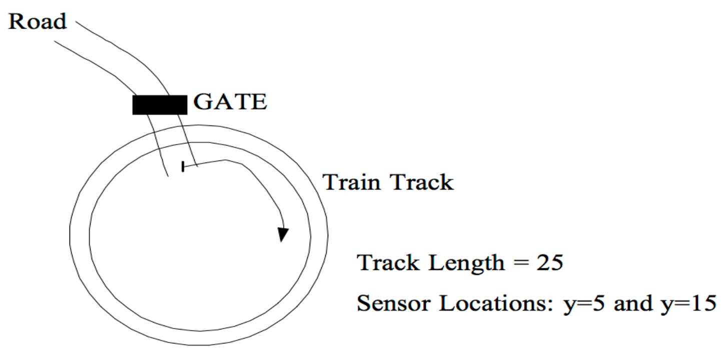

Figure 2: Train Gate-Controller System, Koutsoukos et. al. (2003)

Table 2: Enumeration of all input signals for the train gate-controller problem. s1,…s3 represent generalized signals, where the actual signal name is beside it in parenthesis. Each row of the table is called a "signal vector." The valid bit in the rightmost column is 1 if the signal vector is a valid combination of signals given the constraints of the system, and 0 otherwise.

i1 (sensor1)	i2 (sensor2)	i3 (gate)	valid bit
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0

three signals, *sensor1, sensor2,* and *gate*.

The signals are defined as follows, where *loc* is the location of the train:

Sensor1: is 1 when $5 < loc < 15$ and 0 otherwise.

Sensor2: is 1 when $15 < loc$ modulo $25 < 5$ and 0 otherwise.

Gate: is 1 when DOWN is sent to the gate and 0 when UP is sent to gate.

We now create our table by enumerating all possible value combinations of our three signals, yielding a table with 8 signal vectors.

The valid bits which are red represent states in which *i1 equal to i2,* which are invalid states since both signals cannot both be zero or one. The other valid bits are set based on the constraints of the system. Consider signal vector [0 1 1], the train has passed sensor2 and not sensor1 and thus is nearing the gate, but the gate is UP. Thus, cars may be hit by the train and this is an invalid state, shown by the valid bit of 0. Similar reasoning can be made for the other valid bits.

In real time, if our system ever has a signal vector with a valid bit of 0, we stop the train because our system is invalid. Let's say that the signal vector is [0 1 1] as before. Then we can determine which signal was compromised by toggling each bit until we achieve a valid state (since we knew the system was in a valid state before the attacker compromised, i.e. toggled, the signal). Performing the *Intrusion Detection* algorithm, left as an exercise, we find that only the set when all 3 signals are toggled yields a valid state, thus the attacker must have compromised all three signals and the correct state is [1 0 0].

Consider an example when the current signal vector of the system is [0 1 1], a valid state. We wish to determine the robustness of our system in order to know how many signals must be compromised in order for an intrusion to go undetected. Performing the *Robustness Algorithm,* we first toggle each bit, none of which yield a signal vector with a valid bit, so we do not change *robustArray[0]* or *robustArray[1].* Again, we try toggling every combination of 2 signals, but no resultant signal vector yields a valid bit, so we do not change *robustArray[2].* However, when we toggle 3 signals, we achieve a valid bit, so we set *robustArray[3] = false.* We build our set, robust zone by adding all indices of *robustArray* which are true, in this case, [0 1 2]. Thus, the train gate-controller system is robust against attacks in which 0, 1, or 2 signals are comprised, but will not guarantee to detect an attack if 3 signals are compromised (and in this case, will never detect this attack since there is only one way to toggle all 3 signals and it is a valid bit).

Thus, we have demonstrated that we can determine the robustness of a system as well as guarantee intrusion detection in the domain of our robust zone.

CONCLUSIONS

When considering CPS security, one must consider the affects that CPS security attacks can have on human life as well as the additional security measures introduced by the physical components of the system. Through a survey of examples, we have seen how CPS security attacks can be devastating to both physical objects and human life, validating the need for greater efforts in CPS security. With this in mind, we developed an algorithm which determines the robustness of a CPS, therein providing information which can provide intrusion detection methodologies.

The *robustness algorithm* is very useful for intrusion detection if we know constraints about a system. For example, if we determine the robust zone of a given system is [0 1 2 3] and we know an attacker is constrained to changing no more than 2 signals at a time, than we guarantee intrusion detection, and in the case of such an attack, we can identify which signals may have been comprised.

In general, we can use encryption keys on our signals to ensure that an attacker cannot change two signals at once but can change at most one signal. This algorithm in conjunction with encryption keys ensures we always detect every intrusion as long as our robust zone contains the subset [0 1]. In other words, if the system is robust to a single altered signal, then we guarantee that our system will detect this intrusion. However, two downfalls exist: (1) if the system is found to be robust up to *n* signals, and the attacker compromises *n + 1* signals, we cannot guarantee that that we will detect the attack, since it is possible that the *n + 1* signals are changed to another valid state, when the system is in a different state (e.g. the gate is up when it should be down, but the signals are in a valid state). (2) The intrusion detection algorithm does not provide exact identification as to which signal has been compromised, but instead provides a set of possible compromised signals. However in some cases, such as the train gate-controller example, exact identification is possible.

Above all, this algorithm is highly extensible. In this paper, we discuss the algorithm over the domain of integers, however one could imagine extending this algorithm to support variables over the real domain. In the case of hybrid automata, the signal constraints can be real-valued differential equations, allowing much more advanced applications of the concepts discussed herein.

References

[1] Rajkumar, Ragunathan Raj, et al. "Cyber-physical systems: the next computing revolution." *Proceedings of the 47th Design Automation Conference. ACM*, 2010.

[2] Poovendran, R. A. D. H. A. "Cyber–Physical Systems: Close Encounters Between Two Parallel Worlds [Point of View]." *Proceedings of the IEEE* 98.8 (2010): 1363-1366.

[3] Koscher, Karl, et al. "Experimental security analysis of a modern automobile. "Security and Privacy (SP), *2010 IEEE Symposium on. IEEE*, 2010.

[4] Stallings, William. "Cryptography and network security, principles and practice," 2011, *Prentice Hall*.

[5] Cardenas, Alvaro A., Tanya Roosta, and Shankar Sastry. "Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems." *Ad Hoc Networks* 7.8 (2009): 1434-1447.

[6] Guo, Chuanxiong, Helen J. Wang, and Wenwu Zhu. "Smart-phone attacks and defenses." *HotNets III*. 2004.

[7] Falliere, Nicolas, Liam O. Murchu, and Eric Chien. "W32. stuxnet dossier."*White paper, Symantec Corp., Security Response* (2011).

[8] Morel, Benoît. "Anomaly Based Intrusion Detection and Artificial Intelligence."*InTech Open Book Chapter* (2011): 19-38.

[9] Gagnon, François, Babak Esfandiari, and Leopoldo Bertossi. "A hybrid approach to operating system discovery using answer set programming."*Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*. IEEE, 2007.

[10] Shiva, Sajjan, Sankardas Roy, and Dipankar Dasgupta. "Game theory for cyber security." *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*. ACM, 2010.

[11] Antsaklis, Panos J., and Xenofon D. Koutsoukos. "Hybrid systems: Review and recent progress." *Software Enabled Control: Information Technology for Dynamical Systems. NY: Wiley-IEEE* (2003).