4

EVOLVING SECURITY

A. Sonalker[1] and E. Griffor[2]

[1]STEER Tech, Columbia, MD, United States [2]National Institute of Standards and Technology (NIST), Gaithersburg, MD, United States

4.1 Need for Security in a Cyber-Physical System

The need for security began with four simple questions:

- *Is the claimant who they claim to be?*
- *Is the claimant allowed to do what they intend to do?*
- *Did the data change in transit?*
- *Can the data be kept secret to all except the intended party?*

These four tenets are commonly referred to as authentication, authorization, integrity, and confidentiality, respectively, and are an excellent starting point in laying the ground work for "secure" design. With cyber-physical systems (CPS) the problem begins at this same initial point, except the two end parties could be machine-to-human, human-to-machine, or machine-to-machine and that there exist some nondigital elements of the system (mechanical, hydraulic, pneumatic, nuclear, electrical, etc.). An important aspect of CPS systems, availability, was introduced much later by Needham and Price [1] and comes from reliability engineering. Availability has much to do with fault tolerance and robustness to failure. Any good distributed system is designed to be highly available in the presence of faults [2,3] or fall back to a fail-safe state if it cannot remain available, with or without security in the picture. The chunk of availability attributable to security then is of intrusion tolerance and flow control (predominantly seen in Denial of Service attacks).

CPSs were originally intended to be physical systems with varying levels of automated controls enabled by computation. Control was direct, or remote via a controller network. With the advent of the cyber component and broad connectivity across the Internet of Things (IoT), remote truly became *long-range remote* (off-site). While this transition became a major enabler

Handbook of System Safety and Security. DOI: http://dx.doi.org/10.1016/B978-0-12-803773-7.00004-8

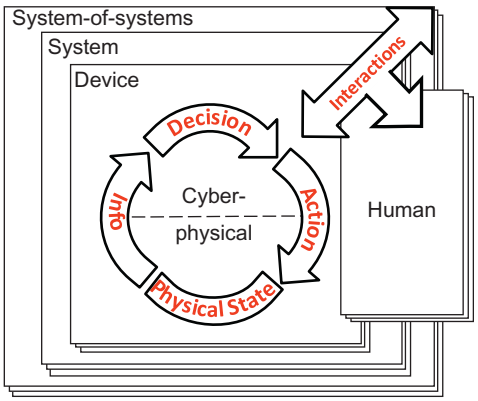

Figure 4.1 Cyber-physical systems.

in making CPSs more efficient, lower cost, as well as easily accessible and controllable, this transition clearly impacted security needs. The questions posed earlier become even more critical, and need to be reassessed, even readdressed. Besides end-point security and credential verification and integrity of the control data, there is a need to ensure that there is *no insertion, fabrication, or replay of legitimate commands*. This aspect is not covered by the four questions, as it happens to be part of the normal and correct functioning of the system. However lack of such a check can result in compromise of control over the assets that comprise many CPSs (Fig. 4.1).

The likelihood of a security breach is a probability determination of *threat capability and vulnerabilities* in the system. Threat capability is a function of accessibility to more sophisticated open source and custom tools, more powerful hardware at lower prices, more knowledge of systems due to the use of more open source software. While there potentially are a few more factors that influence threat capability, our opinion is that the three factors listed above have the highest directly proportional effect on hacker capability, and therefore on the likelihood of breach. Vulnerabilities are directly related to the complexity of the system [4]. The number of lines of code (thousands), i.e., KLOC is a good measure of project size, and therefore software system complexity. A modern automobile, e.g., contains approximately 80–100 million lines of code [5]. Most CPSs today are being driven toward software defined-x, whether it is software-defined networking, software-defined radio, x-by-wire, and more. This is intended for cost reduction, mechanical failure reduction, ease of maintenance compared to hardware maintenance, and even lightweighting (cost and fuel efficiency benefit). As more and more electromechanical

systems are replaced by software, these systems become incrementally complex. Legacy software is often not removed, code is incrementally updated in modules without testing the entire system or understanding interactions with other modules, and testing becomes highly complex and expensive. Defect density (defects per thousand lines of software code), a measurement for software quality, for known systems like Microsoft Windows 10, e.g., is 5 bugs/KLOC, NASA Goddard Flight Systems, which performs very rigorous (very expensive) software testing, has a defect ratio of 0.1 bugs/KLOC [6]. Defects are mainly caused by either design or implementation and can be reduced by rigorous testing. Often times, in a rush to deliver releases, software is not tested as rigorously as needed. While defenders need to find and address most defects, a competent adversary often leverages a single one of these defects (vulnerabilities) to launch an attack and penetrate the system.

4.1.1 Hacker Capability and System Complexity

In Fig. 4.2, *hacker capability* and *system complexity* are graphed for the past, present, and future to convey the changes in trends. As discussed earlier, both system complexity and hacker capability have independently increased over time.

System complexity has increased due to (1) capability enhancement, (2) system expansions, (3) security enhancements, (4) replacing mechanical and hardware functions with software equivalents, (5) incremental development and software complexity associated with leaving legacy codes, and often following suboptimal software engineering practices. As seen above system complexity is directly related to the number of vulnerabilities that can only be mitigated through rigorous testing.

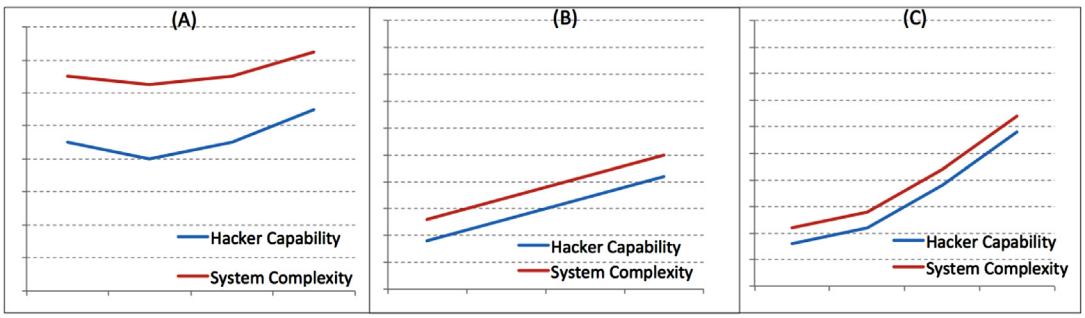

Figure 4.2 Trend in capability versus system complexity as a defense mechanism (A) Past, (B) Present, and (C) Future. STEER Tech.

4.1.2 Security Enhancements to Systems

Take the case of a defense-in-depth model for a CPS. In most cases that I have examined in the automotive domain, system vulnerabilities went from linear to exponential because of the complexity of implementation, as measured by the increased number of lines of software code. While each vulnerability may not directly result in a system breach (access to a core critical function), there are now more vulnerabilities overall. The only reason why each vulnerability does not result in a system breach is because the "gaps" in each layer of the depth model were not directly aligned. Some vulnerabilities are deep within a layer and do not have external exposure, but if an adversary was to penetrate each layer and gain access and control of system resources in each layer, and can then move around in the system, it can find the next vulnerability, expose it and continue to breach deeper into the system. To reiterate, this result has a lot to do with capabilities of the adversary, access to sophisticated tools, and the biggest advantage—increased knowledge of the system. For example, consider what happens when the adversary gets shell access and control and realizes that it is an open source system the adversary is already familiar with. From these graphs, we conclude that the aim of secure system design, in a nutshell, should be to ensure that system defense is always ahead of hacker capability.

4.1.3 Risk Versus Asset Balance

As CPSs become more and more connected, including connectivity for purposes of measurement, control, and transactional purposes, there is a need to revisit traditional security thinking and mechanisms. Obviously, there is a new cost associated with this *change*. This cost can range from additional component costs, to time delays, to process disruption until new mechanisms are streamlined in. Even with new mechanisms there is again a new testing and validation phase. At the very least it will incur additional time, money, and operational delays or, alternately, time-to-market delays in case of production of CPS systems. Therefore to minimize disruption and incremental process cost, we must prioritize. The reassessment of security must be done from a *risks versus assets perspective* keeping this new connectivity and subsystem interplay in mind. Without the latter, the revaluation and reevolution is ineffective. From a risk perspective the questions to ask would be:

- *What are the assets that must be protected at all times?*
- *What kind of "protection" do they need (authentication, authorization, integrity, confidentiality)?*

A threat assessment that begins with identifying the true assets and their protection needs, the six questions, the four initial questions plus these additional two, must be conducted. Once we understand this, then we have to assess what the practical risk factor is given all this new connectivity and the *reach-in* to the assets that were not typically present before. This new reach-in, typically called a *vector* is a particular strategic direction, taken using a particular approach, using a unique path that may either be directly coming in from the outside, or through other subsystems and components.

The point of entry of a vector is called an *entry point*. The set of all exploitable entry points along a logical or physical surface is called an *attack surface*. We will revisit this definition more concretely in the following sections. For example, consider an automobile as our example of a CPS. An external attack surface would be the outer perimeter of the vehicle with all its vulnerable communication entry points. A second attack surface would be an in-cab attack surface, which is still external to the automotive CPSs, but physically inside the car. A third attack surface, now internal to the CPSs, could be the level where noncritical systems meet critical systems, at a *gateway*, for example, that enables communication between subsystems of the CPSs.

A further *security evolution* is that the adversary model needs to be revaluated for a variety of reasons. Systems have evolved beyond single standalone systems; remote monitoring, command and control, and increasing productivity being the chief reasons for increasing the level of automation and connectivity between multiple systems. Traditional adversary models do not take these complex interactions into account. For example, traditional models include:

- byzantine (arbitrary failures),
- natural failures,
- human error,
- economic espionage (economically motivated cybercriminal).

All of these models are meant for nonadaptive interaction between the adversary and the system. They are also not intended for modeling cascading failures due to an external trigger. Most byzantine failures are countered by letting the systems fall back into a fail-safe mode. Fail-safe modes are created for each individual subsystem but rarely for the entire system overall. An *air-gap* is presumed: the notion is that by design, a system in fail-safe mode will do no harm to an adjacent system. Today, we find this assumption to be grossly inadequate. Therefore today systems must be designed to respond to failure, intrusion, or any type of anomaly while keeping transactions, interactions, and system responses “in mind.” An intelligent

adversary can leverage these transactions to bootstrap into a more powerful and undetectable attack. In essence, the system functionality, by design, can be leveraged to attack the system itself. Today, the adversary we face is far more motivated; an adversary whose intent is to subvert the system to the point of critical failure and cause cascading failures of other reliant systems. Today, the most intelligent adversaries aim to exploit systems in a manner that is *consistent with the design and function of the system* but has a disruptive intention. We have seen several examples of this today ranging from cybercriminals to rookie hackers to cyberterrorists and nation states exploiting built-in features used in a manner not intended by the designers of the system, yet fully consistent with the functioning of the system. Power shell control, access through open ports, and gaining root are some of the ways these adversaries initially enter the system. Replaying legitimate messages to increment system state to the point of failure, for example, is another way of exploiting legitimate functions for malicious use. All these are possible because the system designers did not ask the right questions at the time of design, or because the security model was not reevaluated when the system was incrementally updated with new connectivity entry points.

It should be sufficiently clear now why the adversary model used in designing and evaluating cybersecurity in these CPSs must be remodeled, that is, reconsidered and modeled again. While this is true universally for all electronic systems including financial networks and their systems, it is especially important for CPSs due to the intermix of digital electronics with electrical, mechanical, pneumatic, hydraulic, and systems with kinetic components. Traditionally air-gapped systems also need to be revisited because the air gaps may be in one dimension (mechanical, digital, or pneumatic) but may still be penetrable in a different dimension.

In the rest of the chapter, we will revisit adversary strategies and model them. Then we will model attack surfaces from a risk probability perspective, and assign probabilistic risk values to the attack vector paths resulting in a composite risk value for each attack vector. Then we will determine the overall risk posture for a connected system by weighing that risk value against the asset. Finally, we will subject the most critically risky assets in the connected system to "secure design." By this point the design that will emerge will be flexible. The design framework will be flexible and easy to reevaluate when additional system capabilities (and connectivity points) are added over time. We have now created an *evolving security model!*

4.2 New Adversary Modeling

Most CPSs are built with significant fault tolerance via closed loop control systems and/or redundancy. This makes it hard for a single message adversary to be successful. Most successful strategies we have seen involve multiple messages built on top of each other, or involving contextual information that allows the attack to penetrate through the system's defenses.

To model this type of adversary, we make the following assumption:

(AO) The Adversary is adaptive, and "Online."

Before we delve deeper into adversary strategies, it is essential to review the classes of adversaries we have considered.

The oblivious adversary, or the stateless adversary. This is the weakest form of adversary model. This type of adversary can simply send messages (attack packets) to the system but does not know how the system will respond. Each message is in isolation from the next one and does not form a sequence. In that sense, this adversary has the weakest advantage.

The adaptive online adversary. This adversary has an understanding of the system, and its response. It can craft a series of messages, or a comprehensive attack strategy consisting of multiple messages that are contextually relevant and can mimic real messages in pattern and time. This adversary is assumed to be "online" in the sense that it must be real-time in its decision making, and its strategy decisions are made *before* the system response. The next decision, however, can be made based on the previous decision or the response of the system. In this sense, this adversary has medium advantage. (Greater than the oblivious adversary, but lesser than the adaptive offline adversary.)

The adaptive offline adversary. This adversary not only has the ability to build and launch sophisticated multimessage attacks like the previous adversary, but it can also build all its strategies offline. The implied assumption here is that in order to do so, it has access to the data dictionary of the system, any and all relevant cryptographic primitive generators, and all timing/pattern information. In that sense this is an extremely powerful adversary. Most insider attackers and nation state attackers would fall into this category.

An adaptive offline adversary is less likely to be our main and most probable adversary. Similarly the oblivious adversary is also very less likely to be one we need to spend considerable resources to defend. The medium adversary, that is, the adaptive online adversary is the most commonly encountered, and most likely threat to connected CPS systems.

For the rest of the chapter, we will base our work on the adaptive online adversary. However, we will also show how modifications can be made to model the adaptive offline adversary for the most critical scenarios that do face nation state threats (e.g., Electric Grid).

If M_1 is the first message sequence in the *adaptive online adversary strategy* S_1, and R_1 is the first system response sequence to that first adversary message sequence, then we can describe the strategy and *causal chain* in a clear way. A general interaction between the adversary and the system has the form:

Interaction: I: $M_1R_1M_2R_2 \ldots .$

Thus the general form of an *adversarial interaction* is a sequence of messages sent by the adversary, followed by a sequence of system responses, followed by a sequence of adversary messages and then a sequence of system responses, and so on.

For example, $M_1 = m_1.m_2.m_3 \ldots m_n$

that is, M_1 consists of all messages $m_1{:}m_n$ that are sent before the system response R_1.

Similarly, R_1 can be a single message response r_1, or a series of response messages r_1: r_n.

Note that because M_2 is dependent on R_1 therefore, R_1 becomes a part of adversary strategy S_1, hence the *online dependency.*

Adversaries can create multiple strategies that involve getting the system to respond (R) and use that response as the adversary's next step in the attack strategy. Each of the attack messages and their responses will have a probabilistic risk value associated with it in our model. These are used to "add up" the risk probability associated with an attack strategy and is ultimately used to model an appropriate protection strategy.

4.2.1 Attack Surfaces

For each *threat*, there exist several attack surfaces with multiple entry points per attack surface. Each type of entry being an attack vector. It would be exhaustive to model each and every one of them. However, we can generalize and create *rules of thumb* that would be useful for understanding how systems can be impacted by attack vectors. Further, this can facilitate evaluation of the risk associated with a given entry point, and demonstrate how interactions between interconnected systems, modules, or software routines can increase/decrease risk. Finally, we can use it to model how connectivity impacts risk.

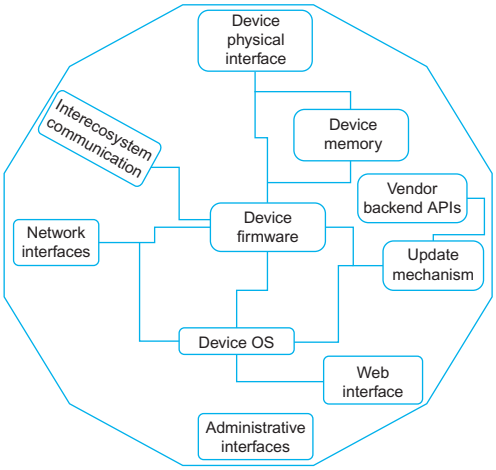

Figure 4.3 Attack surface, and typical entry points.

4.2.2 Weighting the Attack Surfaces

In the colloquial sense the phrase *exterior attack surface* refers to the exterior, penetrable boundary of a CPS system in its entirety; a boundary that an adversary can connect directly with. Components controlling critical functions may not necessarily have an obvious connection to the exterior surface. However there is risk associated with the malfunction or arbitrary failure of a component that can lead to catastrophic loss. Most risk assessment exercises capture this component of risk well. However a thought must be lent here to the fact that these critical components are connected to, and interact with other components. Therefore, there must be some risk associated with interactions with another component that could lead to failures. This, in other words, becomes the "attack surface" of the internal component! Extrapolating this further, we see that one is likely to find a path from the exterior attack surface to the interior attack surface of a critical component sometimes by sheer perseverance, but often by stacking known vulnerabilities and exploits *from the outside in*. Often there may be multiple paths from the exterior surface to the interior through different components.

Therefore a true risk assessment must be able to capture and calculate the risk arising from an exterior attack surface for a component that is buried deep within the system. This risk assessment should capture the different paths that one may take to get to the interior systems and their "penetrability."

4.2.3 Attack Entry Points

For each attack surface, and a given threat vector, we enumerate each entry point. For a CPS, it begins at each point with which an adversary can interact, irrespective of access control. On a software level this translates to an entry point that can be accessed through global identifiers, handlers, input strings, and public methods to name a few. On a network level, open ports, socket identifiers, other communication handles are the main entry points to that module. These entry points will allow an adversary to gain access commensurate with its access privileges.

Formally, we enumerate the attack entry points associated with an attack surface as a sequence $e_1{:}e_n$ where each lower case e is an element of E, the set of all attack entry points.

4.2.4 Role-Based Access

Most systems today have varying degrees of access control. Access control for multiuser systems is used to limit the ability of users from accessing all parts of the system. Users, as defined earlier, for CPS systems can be humans, machines, interprocess communication routines, and more. Role-based access control reduces the breadth of entry points, but increases the depth of each entry point for an interconnected system. Therefore role (access control) is an important parameter in the risk assessment equation. Similarly, resources that can be accessed via entry points (subject to role-based access) are a third important parameter in risk assessment.

If any entry point e_1 is limited by access control (role-based only), then we assign a weight w_1 to it. This weight has a probabilistic component to it that is determined by examining the underlying system. For systems that are greater than one-hop away from the exterior surface weight w is calculated for each entry point, and combined with the probabilistic risk factor of the upstream component/system. These calculations can be done on a per-system down to a per-component basis based on need and relevance.

$$\text{Vector for entry point } e_1 = e_{1j} \cdot w_{1j}.$$

If there exist multiple ways of entering the same entry point (e.g., as in the case with Bluetooth stacks, or OBD-II in case of a vehicle) then

$$\textit{Entry point risk } \mathscr{R}_i = \sum e_{ij} w_{ij}$$

where the second index j represents the subsystem from where the vector originates. A two index entry point therefore

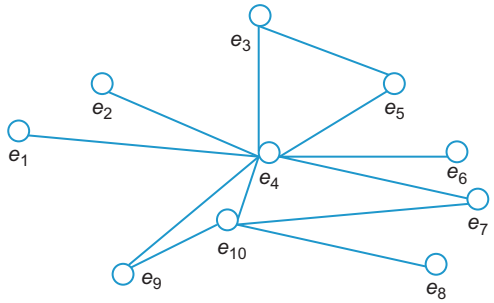

Figure 4.4 Connected subsystem graph.

denotes an entry point internal to the CPS but outside the subsystem. As discussed earlier, an internal entry point (often a critical system) can be reached by multiple entry points. That is, there exist multiple vectors to reach an internal entry point e_k (where k belongs to all n subsystems). Then, e_k is the sum of all vectors leading into it. This can now be represented by a graph. Taking the example of Fig. 4.3 further, we represent it as a graph in Fig. 4.4.

Then, *risk for* $e_{10} = e_4 w_4 + e_7 w_7 + e_8 w_8 + e_9 w_9$

While $e_4 = e_{1:3} w_{1:3} + e_{5:7} w_{5:7}$, and so on.

Cyclic graphs will need to be resolved, and directionality will need to be weighted separately (if there exists any). This topic can take a whole course of its own, and if readers are interested there are several graph theoretical concepts that can be applied to reduce risk for e_4 and e_{10} (and any system under consideration in general). Readers are encouraged to pursue independently as this depth would be outside the scope of this chapter.

4.2.5 Resource Access

The resources that can be accessed by an adversary from each entry point must be identified. For a CPS, resources can be data, databases, cryptosystems, access control libraries, meta-data, services, computation power, memory, network channels, user privileged information, credentials, and more. All resources must be attributed a priority value that reflects the damage that results from the loss of control over the resource. A loss of control of a resource is a *gain* for the adversary. The adversary may use resources gained to further penetrate the system and cause more damage. Clearly, for an interconnected system, resource risk multiplies as the attacker penetrates the system further. Similarly, connectivity increases the resource risks *if* there exists a path to the resource from the exterior

surface that is a combination of penetrating internal attack surfaces, *adaptive* access control, and *adaptive* resource capture.

Formally resources are represented as $g_1{:}g_n$ where G is the overall system (as a resource). The goal of an adversary is to gain access to a coveted resource. The goal of a systems security professional is to protect coveted resources. Coveted resources have higher risk. Both systems and resources can be captured. However captured resources can be leveraged to reduce penetration effort in subsequent system captures (Adaptive Strategy).

Therefore the overall game of the adversary becomes—how to strategize to maximize overall gain of system resources while minimizing effort (optimal RoI). Part of the strategy is to pick the right entry points with the shortest/fastest vector to get to a node with gainful assets. To realize this attack strategy once potential entry points have been selected and the attacker establishes a best guesstimate of the vector path, the attacker will construct appropriate messages, factor in responses and build the attack strategy S.

This is how we would connect all the concepts we have laid out so far.

The role of the cybersecurity practitioner and system designers is to make entry point access harder (increase adversary effort to lower RoI), decrease the likelihood of penetration of each system, lengthen the attack vector path to increase adversary cost (e.g., by increasing partitioning between systems), and maximize protection strategies around gainful assets. This can be done using a combination of hardware and software security techniques and primitives. Mixing hardware (and even physically derived primitives) can be a very high return countermeasure as an adversary now has to find a way to bridge the gap between digital and physical systems (or between digital cyber and digital secure hardware).

4.3 "Connected" System Security Modeling

Now that we can specify system components, entry points, and the likelihood of penetration through an entry point, we can begin modeling connected systems.

We have seen two examples of connected systems thus far. Fig. 4.5 illustrates digitally interconnected subcomponents that pass data and control through them. Fig. 4.6 illustrates interconnected components that are potentially air gapped over one physical medium but may be connected via another. We will use the earlier figure for the purpose of modeling connected systems security.

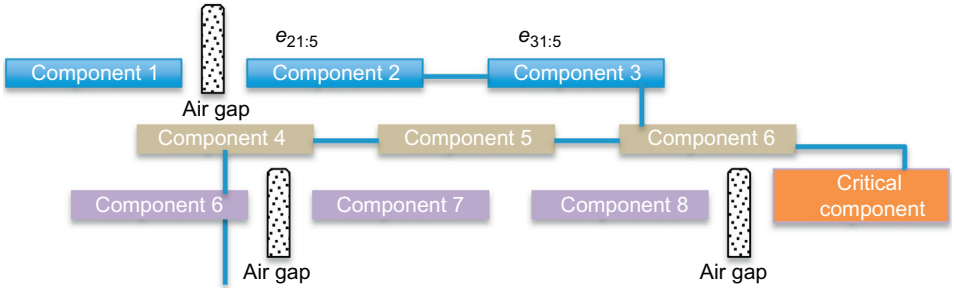

Figure 4.5 Generalized cyber-physical system from an interconnected composable perspective.

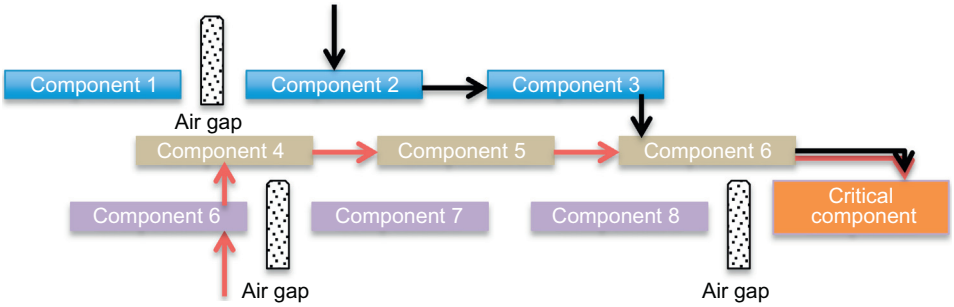

Figure 4.6 Directional impact on risk assessment and countermeasure design.

The best place to start would be at nodes that have no upstream, only downstream connections into the system under consideration. Typically, these are external facing points, and one can easily see that these will be potential attack entry points. Referring back to our example Figure 4.1, e_1, e_2, e_3, e_7, e_8, e_9, are external facing interfaces (Fig. 4.6).

We calculate the likelihood of penetration of each entry point. This is a security analytic operation in itself. We will find that the likelihood follows a Pareto distribution. Let us assign a symbolic notation w to each of them.

Therefore, risk at e_4

$$\mathcal{R}e_4 = e_1w_1 + e_2w_2 + e_3w_3 + e_5w_5 + e_6w_6 + e_7w_7,$$

Now, e_4 happens to have resources. If penetrated and e_4 is captured, then the adversary will have access to all the resources that e_4 has. The collective shared resources and private resources at e_4 is g_4. For example, here e_4 shares memory resources with e_5 and e_{10}, so part of g_4 comes from e_5 and e_{10}'s resources.

The aim of the adversary is to maximize gain g_4 while minimizing its effort leading into e_4. In other words the adversary needs to find the shortest, least resistance path to e_4 among e_1 to e_9.

Now, while the penetration itself has a likelihood of being successful, the adversary's choice of which entry point and what technique is not. It is a carefully crafted decision after assessment. Therefore, we use Wald's min max model [7] to pick decisions and create strategies that the adversary will use to play off the system. Note that we are using the adaptive online adversary which implies that the adversary builds its strategy by choosing messages, then letting the system respond and building on top of the response. The Wald's min max model suits this adaptive online adversary model very well because it is a nonprobabilistic, robust decision-making model in which the optimal decision (which is the overall intention for an optimal RoI) is one whose worst outcome is at least as good as the worst outcome in any other scenario.

Mathematically,

$$f^* = \max_{s \in \mathscr{S}} \min_{M \epsilon \rho(S)} f(s, M)$$

where $\mathscr{S}$ is a set of alternate decisions, actions or strategies, $\rho(S)$ is the set of states associated with actionable message M, and $f(s, M)$ is the return of a strategy that takes place in the state of s.

In this manner a strategy S is selected and executed by an adversary that comprises of actionable messages M.

At the next node the adversary's advantage is evident as

$$f^* = \max_{s \in \mathscr{S}} \mathrm{g}_{\mathrm{n}=4}. \min_{M \epsilon \rho(S)} f(s, M)$$

In other words, for interconnected systems, the likelihood of success for an adversary, as the play develops increases since the adversary collects more resources (gains) as he proceeds.

4.4 Directional Threat Assessment

So far we have focused on external attack surfaces and the probability of an attacker penetrating the external surface to make their way *in* to a critical system (or component). Most often times, system designers miss the fact that attackers can exist on the inside of a system and work their way out. They either get in via a *physical* security lapse or, are *insiders* to begin with. This has been the predominant modus operandi in a vast majority of the successful high-profile attacks on CPSs we have seen recently [8–11]. Even if systems are built with more resistance to outside threat, they are built with almost no resistance going from inside out. As a result, the threat vector is

asymmetric in one direction; there is significantly less work to be done to penetrate from the inside out.

Threat assessment ultimately should be *directionally oriented* because of this asymmetry.

Since we are holistically dealing with security evolution, measuring cybersecurity risk in CPSs with the intention of being able to prioritize, assign resources, and design countermeasures commensurate with risk, we must consider this type of attack as well. Additionally, insider attacks and IoT attacks fall into this category. Both types of attacks are the most common and most destructive types of attacks known today. And rightly so, most systems lack defenses against insider threats and IoT threats.

4.5 Big Picture CPS Systems—IoT

The approach presented in this chapter to new age online adversary modeling, modeling systems as a graph of interconnected nodes complete with individual resources and capabilities that can be conquered and leveraged, can with some expertise be applied at several different levels of abstraction. In this chapter we presented it for a single CPS system, say for example an automobile or a scada system. We modeled interactions, entry points, probabilities of being subverted, and sequential strategies at a single CPS level. By changing the level of abstraction from a single system point of view to larger systems point of view, we can model highly diverse systems of systems like those typical of the *IoT ecosystem*. Everything in the IoT is a CPS system. Therefore, by abstracting a given system as a larger IoT system, and then decomposing it into its individual CPS systems, one can model IoT security using the approach shown in this chapter. One can model an IoT adversary's strategies and moves, and upon successful automation of adversarial strategies, one can even create automated defenses for IoT.

4.6 Conclusion

In the course of this chapter, we began an honest discussion from the fundamental tenets of security and how they evolved from those for closed systems to those appropriate for connected cyberphysical systems. The right questions that were asked to visualize system behavior at system boundaries and model them as discrete interactions allowed us to formally specify these highly complex systems at their various attack surfaces. Correctly realizing an evolved adversary, at its full capabilities and intelligence more powerful than the typical offline

adversary, enabled us to formally model and analyze the full extent of the damage an evolved adversary is capable of exerting. With a clear understanding of a system's capabilities, vulnerabilities, and available resources, we could then realize the full extent of protective measures needed and understand where they need to be localized for maximum protection and ROI. The novel use of the combination of graph theory to model a system, subsystem connections, and resources that can be leveraged, as they are owned and acquired along with powerful adaptive game theory to model adversary and defense strategies, allows us to automate the defense process. As the *game* develops further, our defense now has the intelligence to calculate what resources are now lost to adversarial control and the best strategic moves to be made, under ever changing circumstances, in order to mount a defense. In other words, we can create *autonomous defense systems* fully capable of detecting advanced persistent threats and making their best attempt to automatically respond and recover from them, i.e., the evolution of security from static authentication systems and cryptosystems to fully autonomous computing systems capable of automated defense. The future will simply be more automated and more autonomous intelligent system security.

References

[1] Geraint Price, The interaction between fault tolerance and security, Technical Report Number 479, UCAM-CL-TR-479, ISSN 1476-2986 Cambridge, UK, 1999.

[2] T. Anderson, P.A. Lee, Fault Tolerance: Principles and Practice, Prentice-Hall International, Englewood Cliffs, NJ, 1981.

[3] T. Anderson, P.A. Lee, S.K. Shrivastava, A model of recoverability in multilevel systems, IEEE Trans. Softw. Eng SE-4 (6) (1978) 486–494.

[4] V. Mishra, H. Dion, Y. Bar-Yam, Vulnerability analysis of high dimensional complex systems, in: S. Dolev, J. Cobb, M. Fischer, M. Yung (Eds.), Stabilization, Safety, and Security of Distributed Systems, Springer-Verlag, Berlin, 2010.

[5] Battelle, NEM–Automotive Intrusion detection system, Columbus, OH. <http://www.battelle.org/newsroom/press-releases/new-system-detects-and-alerts-to-automobile-cyber-attacks>, 2014.

[6] R. Stephens, Testing, Beginning Software Engineering (Chapter 8), John Wiley and Sons Publishing, 2015, p. 174.

[7] A. Wald, Statistical Decision Functions, Wiley, New York, NY, 1950.

[8] T. Maiziere, The IT Security in Germany in 2014 Publication, in: *Die Lage der IT-Sicherheit in Deutschland*, December 2014.

[9] R. Lee M. Assante, T. Conway, German Steel Mills Attack, in: SANS ICS Defense Control Use Case, Case Study 2, December 2014.

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

[11] A. Greenberg, Hackers remotely kill a Jeep Cherokee on the highway—with me in it, Wired Magazine (Online edition) (July 2015).