From Cyber Situational Awareness to Adaptive Cyber Defense: Leveling the Cyber Playing Field

Massimiliano Albanese(✉)

George Mason University, Fairfax, VA, USA
malbanes@gmu.edu

Abstract. In the cyber security landscape, the asymmetric relationship between defender and attacker tends to favor the attacker: while the defender needs to protect a system against all possible ways of breaching it, the attacker needs to identify and exploit only one vulnerable entry point in order to succeed. In this chapter, we show how we can effectively reverse such intrinsic asymmetry in favor of the defender by concurrently pursuing two complementary objectives: increasing the defender's understanding of multiple facets of the cyber landscape – referred to as Cyber Situational Awareness (CSA) – and creating uncertainty for the attacker through Moving Target Defense (MTD) or Adaptive Cyber Defense (ACD) techniques. This chapter provides a brief overview of contributions in these areas, and discusses future research directions.

1 Introduction

In the cyber security landscape, the relationship between defender and attacker is typically asymmetric and tends to disproportionally favor the attacker, as the defender needs to protect a system against all possible ways of breaching it, whereas the attacker has to identify and exploit only a single vulnerable entry point in order to succeed. The notional diagram of Fig. 1 shows the relationship between the attacker's effort and the defender's effort over time. Although the required effort may fluctuate over time for both the attacker and the defender, the attacker consistently maintains an advantage over the defender.

In order to limit the attacker's advantage, and potentially level the cyber playing field, we argue that two objectives must be pursued concurrently. On one side, to increase operational efficiency and reduce the defensive effort, we need to improve the defender's understanding of multiple facets of the cyber landscape through Cyber Situational Awareness (CSA) techniques [16]. On the other side, to increase the attacker's effort, we need to create uncertainty about information on the target system, which the attacker may have gathered over time, through Moving Target Defense (MTD) or Adaptive Cyber Defense (ACD) techniques [11]. The diagram of Fig. 2 shows how the deployment of CSA and ACD techniques can significantly reduce the gap between attacker's and defender's effort.

P. Samarati et al. (Eds.): Jajodia Festschrift, LNCS 11170, pp. 1–23, 2018.
https://doi.org/10.1007/978-3-030-04834-1_1

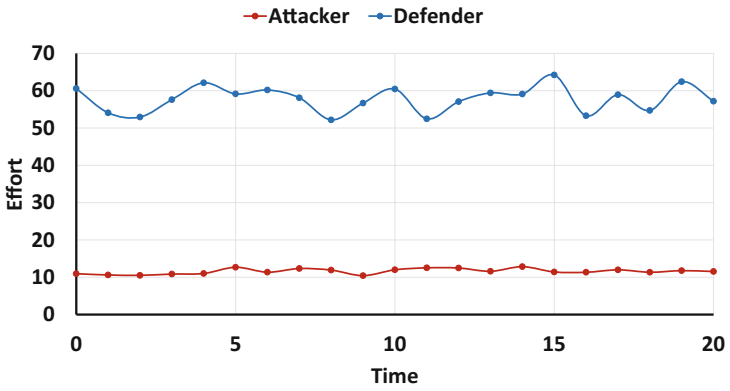

Fig. 1. Attacker's effort vs. defender's effort in a typical scenario, before deploying CSA and ACD mechanisms

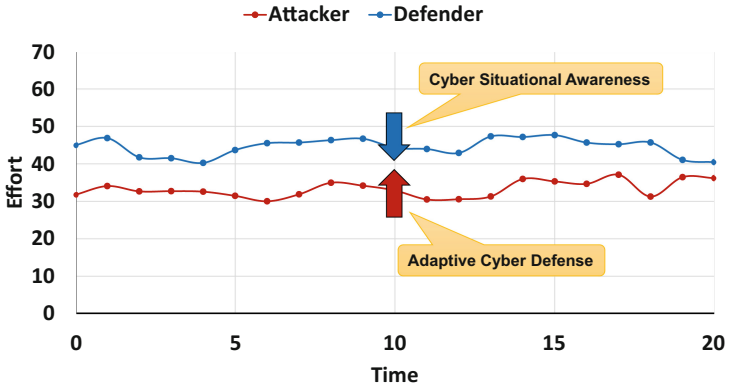

Fig. 2. Impact of CSA and ACD on reducing the gap between attacker's and defender's effort

Current research in these relatively new areas has shown promise to significantly enhance our defensive capabilities. However, much work remains to be done as we aim to push our CSA and ACD capabilities beyond simply leveling the cyber playing field, so as to completely reverse the intrinsic asymmetry of today's cyber security landscape in favor of the defender, as shown in the notional diagram of Fig. 3.

This chapter provides an introduction to the fields of Cyber Situational Awareness and Adaptive Cyber Defense, and a brief overview of contributions in these areas resulting from the author's collaboration with Dr. Jajodia.

The remainder of this chapter is organized as follows. Section 2 introduces the notion of Cyber Situational Awareness, along with a practical motivating example, and describes several key contributions in this area. Similarly, Sect. 3 introduces Cyber Situational Awareness and describes several key contributions.

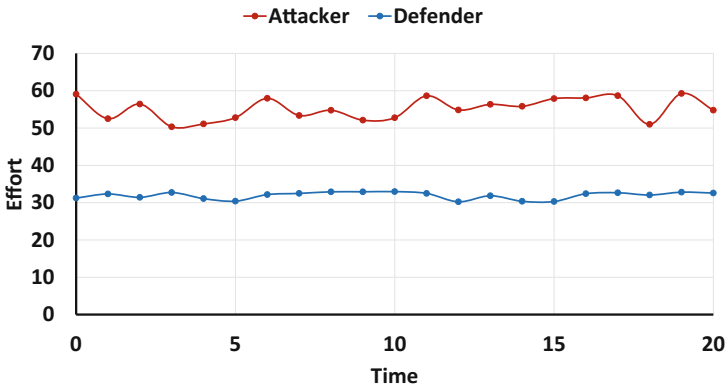

Fig. 3. Long-term objective: reversing the asymmetric relationship between defender and attacker

Finally, Sect. 4 provides some concluding remarks and indicates possible future research directions.

2 Cyber Situational Awareness

Without loss of generality, the process of situational awareness can be viewed as a three-phase process: situation perception, situation comprehension, and situation projection [2]. *Perception* provides information about the status, attributes, and dynamics of relevant elements within the environment. *Comprehension* of the situation encompasses how people combine, interpret, store, and retain information. Finally, *Projection* of the elements of the environment (situation) into the near future entails the ability to make predictions based on the knowledge acquired through perception and comprehension.

In order to make informed decisions, security analysts need to acquire information about the current situation, the impact and evolution of ongoing attacks, the behavior of attackers, the quality of available information and models, and the plausible futures of the current situation. Collectively, this information contributes to the process of forming cyber situational awareness.

In this section, we describe several techniques, mechanisms, and tools that can help form and leverage different types of cyber situational awareness. These capabilities are presented as part of a comprehensive framework that aims at enhancing traditional cyber defense by automating many of the processes that have traditionally required a significant involvement of human analysts. Ideally, we envision the evolution of the current human-in-the-loop approach to cyber defense to a human-on-the-loop paradigm, where human analysts would only be responsible for validating or sanitizing the results generated by automated tools, rather than having to comb through daunting amounts of log entries and security alerts.

Currently, a security analyst plays a major role in all the operational aspects of maintaining the security of an enterprise. Security analysts are also responsible for studying the threat landscape with an eye towards emerging threats. Unfortunately, given the current state of the art in the area of automation, the operational aspects of IT security may still be too time-consuming to allow this type of outward-looking focus in most realistic scenarios. Therefore, the scenario we envision – where automated tools would gather and preprocess large amounts of data on behalf of the analyst – is a highly desirable one. In the following, we define the fundamental questions that, ideally, an effective Cyber Situational Awareness framework should be able to automatically answer. For each question, we identify the inputs as well the outputs of the Cyber Situational Awareness process.

1. **Current situation.** *Is there any ongoing attack? If so, what resources has the attacker already compromised?*
   Answering this set of questions implies the capability of effectively detecting ongoing intrusions, and identifying the assets that might have been already compromised. With respect to these questions, the input to the CSA process consists of IDS logs, firewall logs, and data from other security monitoring tools. On the other hand, the product of the CSA process is a detailed mapping of current intrusions.
2. **Impact.** *How is the attack impacting the organization or mission? Can we assess the damage?*
   Answering this set of questions implies the capability of accurately assessing the impact of ongoing attacks. In this case, the CSA process requires knowledge of the organization's assets along with some measure of each asset's value. Based on this information, the output of the CSA process is an estimate of the damage caused so far by ongoing intrusions.
3. **Evolution.** *How is the situation evolving? Can we track all the steps of an attack?*
   Answering this set of questions implies the capability of monitoring ongoing attacks, once such attacks have been detected. In this case, the input to the CSA process is the situational awareness generated in response to the first set of questions above, whereas the output is a detailed understanding of how the attack is progressing. Developing this capability can help *refresh* the situational awareness formed in response to the first two sets of questions and maintain it current.
4. **Behavior.** *How are the attackers expected to behave? What are their strategies?*
   Answering this set of questions implies the capability of modeling the attacker's behavior in order to understand goals and strategies. Ideally, the output of the CSA process with respect to this set of questions is a set of formal models (e.g., game theoretic or stochastic models) of the attacker's behavior. The attacker's behavior may change over time, therefore models need to adapt to a changing adversarial landscape.

5. **Forensics.** *How did the attacker reach the current state?*
   Answering this question implies the capability of analyzing logs *after the fact* and correlating observations in order to understand how an attack originated and evolved. Although this is not strictly necessary, the CSA process may benefit, in addressing this question, from the situational awareness gained in response to the fourth set of questions. In this case, the output of the CSA process includes a detailed understanding of the weaknesses and vulnerabilities that made the attack possible. This information can help security engineers and administrators harden system configurations in order to prevent similar incidents from occurring again in the future.
6. **Prediction.** *Can we predict plausible futures of the current situation?*
   Answering this question implies the capability of predicting possible moves an attacker may make in the future. With respect to this question, the input to the CSA process consists of the situational awareness gained in response to the first, third, and fourth sets of questions, namely, knowledge about the current situation and its evolution, and knowledge about the attacker's behavior. The output is a set of possible alternative scenarios that may materialize in the future.
7. **Information.** *What information sources can we rely upon? Can we assess their quality?*
   Answering this set of questions implies the capability of assessing the quality of the information sources all other tasks depend upon. With respect to this set of questions, the goal of the CSA process is to generate a detailed understanding of how to weight all different sources when processing information to answer all other sets of questions. Being able to assess the reliability of each information source would enable automated tools to attach a confidence level to each finding.

It is clear from our discussion that some of these questions are strictly correlated, and the ability to answer some of them may depend on the ability to answer other questions. For instance, as we have discussed above, the capability of predicting possible moves an attacker may take depends on the capability of modeling the attacker's behavior. A cross-cutting issue that affects all other aspects of the CSA process is *scalability*. Given the volumes of data involved in answering all these questions, we need to define approaches that are not only effective, but also computationally efficient. In most circumstances, determining a good course of action in a reasonable amount of time may be preferable to determining the best course of action, if this cannot be done in a timely manner.

In conclusion, the situational awareness process in the context of cyber defense entails the generation and maintenance of a body of knowledge that informs and is augmented by all the main functions of the cyber defense process [2]. Situational awareness is generated or used by different mechanisms and tools aimed at addressing the above seven classes of questions that security analysts may routinely ask while executing their work tasks.

2.1 Motivating Example

Throughout this section, we will often refer to the network depicted in Fig. 4 as a motivating example. This network offers two public-facing services, namely *Online Shopping* and *Mobile Order Tracking*, and consists of three subnetworks separated by firewalls. The first two subnetworks implement the two core services, and each of them includes a host accessible from the Internet. The third subnetwork implements the internal business logic, and includes a central database server. An attacker who wants to steal sensitive data from the main database server will need to breach multiple firewalls and gain privileges on several hosts before reaching the target.

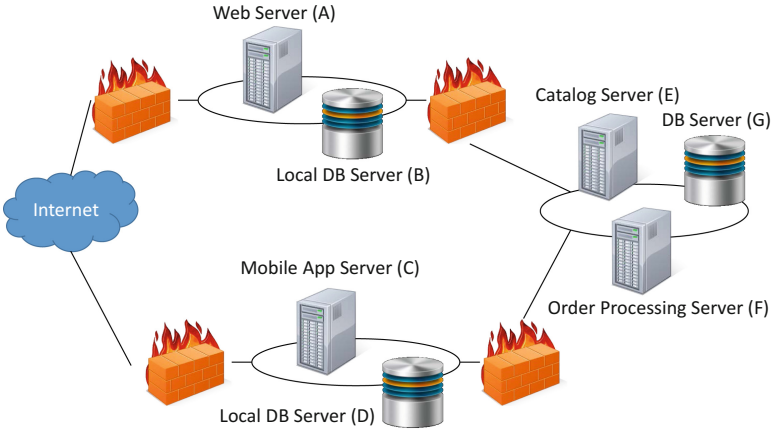

Fig. 4. Motivating example: enterprise network offering two public-facing services

As attackers can leverage the complex interdependencies of network configurations and vulnerabilities to penetrate seemingly well-guarded networks, in-depth analysis of network vulnerabilities must consider attacker exploits not merely in isolation, but in combination. For this reason, we rely on attack graphs to study the vulnerability landscape of any enterprise network. Attack graphs can reveal potential threats by identifying paths that attackers can take to penetrate a network [18].

A partial attack graph for the network of Fig. 4 is shown in Fig. 5. It shows that, once a vulnerability V_C on the Mobile Application Server (host h_C) has been exploited, we can expect the attacker to exploit either vulnerability V_D on host h_D or vulnerability V_F on host h_F. However, the attack graph alone does not answer the following important questions: Which vulnerability has the highest probability of being exploited? Which attack path will have the largest impact on the two services that the network provides? How can we mitigate the risk? Our framework is designed to answer these questions efficiently.

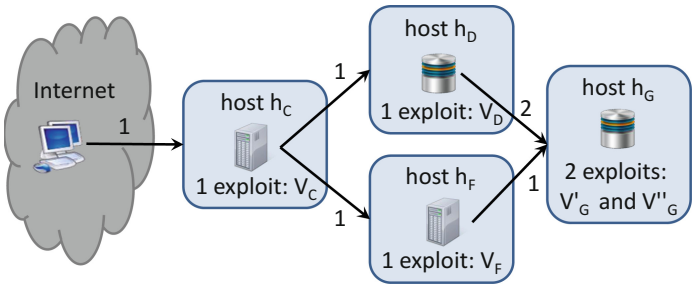

Fig. 5. Partial attack graph for the network of Fig. 4

2.2 The Cyber Situational Awareness Framework

Our Cyber Situational Awareness framework is illustrated in Fig. 6. We start from analyzing the topology of the network, its known vulnerabilities, and possible zero-day vulnerabilities – which must be hypothesized. Vulnerabilities are often interdependent, making traditional point-wise vulnerability analysis ineffective. Our topological approach to vulnerability analysis allows to generate accurate attack graphs showing all the possible attack paths within the network.

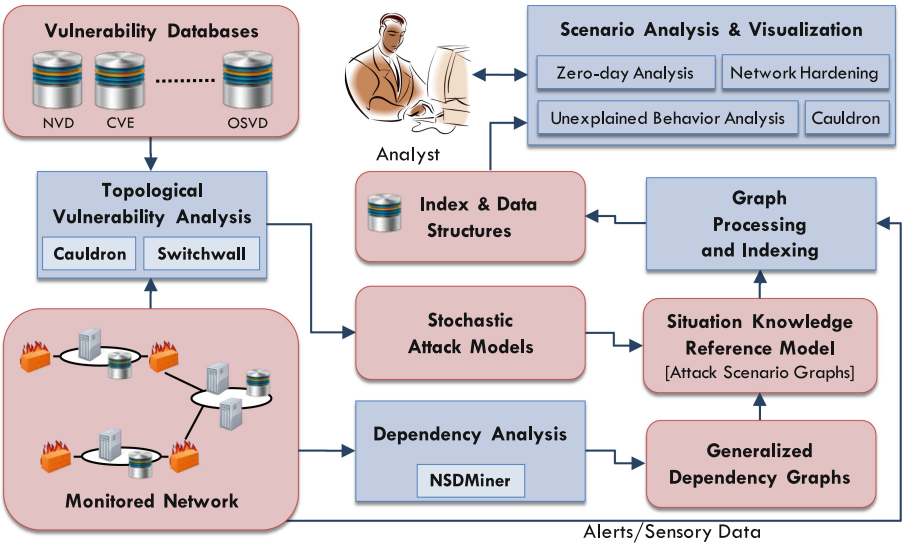

Fig. 6. The Cyber Situational Awareness Framework

A node in an attack graph represents – depending on the level of abstraction – an exploitable vulnerability (or family of vulnerabilities) in either a subnetwork,

an individual host, or an individual software application. Edges represent causal relationships between vulnerabilities. For instance, an edge from a node V_1 to a node V_2 represents the fact that V_2 can be exploited after V_1 has been exploited.

We also perform dependency analysis to discover dependencies among services and hosts and derive dependency graphs encoding how these different network components depend on one another. Dependency analysis is critical to assess current damage (i.e., the value or utility of services disrupted by ongoing attacks) and future damage (i.e., the value or utility of additional services that will be disrupted if no action is taken). In fact, in a complex enterprise, many services may rely on the availability of other services or resources. Therefore, they may be indirectly affected by the compromise of the services or resources they rely upon. Several techniques and tools have been developed to automatically discover dependencies between network services and system components, including the Network Service Dependencies Miner (NSDMiner), which discover dependencies by analyzing passively collected network traffic [22].

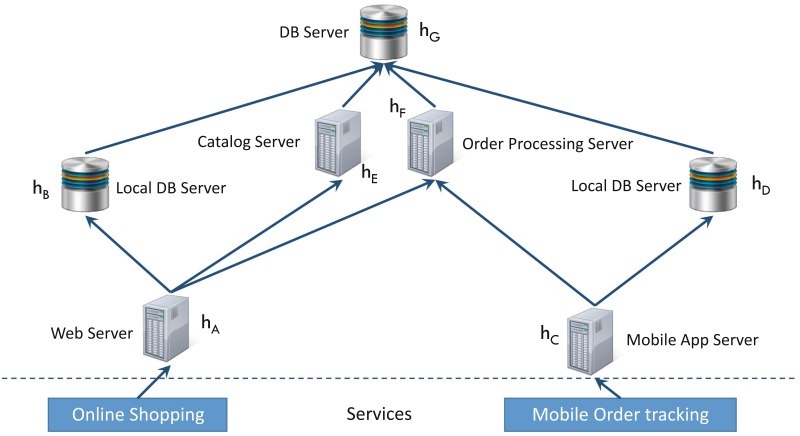

Fig. 7. Dependency graph for the network of Fig. 4

The dependency graph for the network of Fig. 4 is shown in Fig. 7. This graph shows that the two services *Online Shopping* and *Mobile Order Tracking* rely upon hosts h_A and h_C respectively. In turn, host h_A relies upon local database server h_B and host h_E, whereas host h_C relies upon local database server h_D and host h_F. Similarly, h_B, h_D, h_E, and h_F rely upon database server h_G, which appears to be the most critical resource.

By combining the information contained in the dependency and attack graphs in what we call the *attack scenario graph*, we can estimate the future damage that ongoing attacks might cause for each plausible future of the current situation. In practice, the proposed attack scenario graph bridges the semantic gap between known vulnerabilities – at a lower abstraction level – and the missions or services – at a higher abstraction level – that could be ultimately affected

by the exploitation of such vulnerabilities. The attack scenario graph for the network of Fig. 4 is shown in Fig. 8. In this figure, the graph on the left is a complete attack graph modeling all the vulnerabilities in the system and their relationships, where the basic attack graph has been extended to capture probabilistic knowledge of the attacker's behavior as well as temporal constraints on the unfolding of attacks [4,19]. We refer to this class of attack graphs as *probabilistic temporal attack graphs*. Instead, the graph on the right is a dependency graph capturing all the explicit and implicit dependencies between services and hosts, where the two public-facing services have been denoted as h_S (Online Shopping) and h_T (Mobile Order Tracking) respectively. The edges from nodes in the attack graph to nodes in the dependency graph indicate which services or hosts are directly impacted by a successful vulnerability exploit, and are labeled with the corresponding exposure factor, that is the percentage loss the affected asset would experience upon successful execution of the exploit.

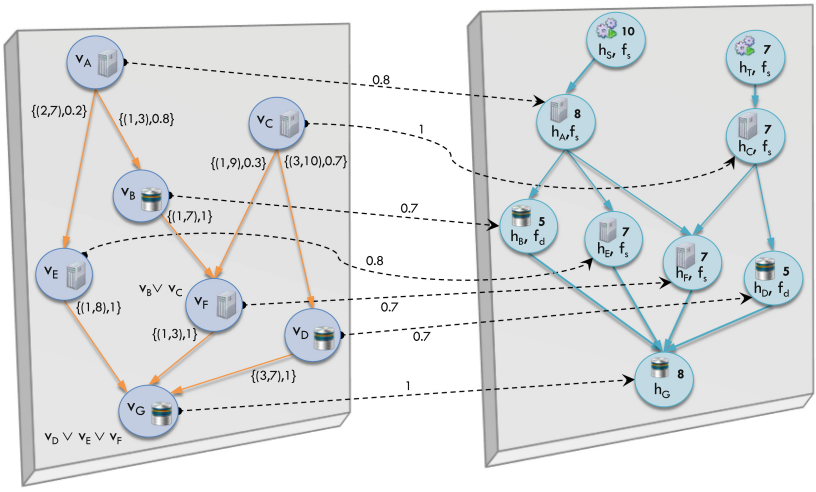

Fig. 8. Attack scenario graph for the network of Fig. 4

In order to address the scalability issues mentioned earlier, we developed novel graph-based data structures and algorithms to enable real-time mapping of alerts to attack graphs and other data analysis tasks. Building upon these graph models, we developed a suite of additional capabilities and tools, including topological vulnerability analysis [13], network hardening [5], and zero-day analysis [7], which we discuss in the following subsections.

In summary, this framework can provide security analysts with a high-level view of the cyber situation. From the simple example of Fig. 8 – which models a system including only a few hosts and services – it is clear that manual analysis could be extremely time-consuming even for relatively small systems. Instead, the tools that make up this framework provide analysts with a better

understanding of the situation, thus enabling them to focus on higher-level tasks that require experience and intuition, and thus more difficult to automate. For instance, the framework could automatically generate a ranked list of recommendations on the best course of action analysts should take to minimize the impact of ongoing and future attacks. Then, analysts may leverage their experience and intuition to select the best course of action amongst those proposed.

Topological Vulnerability Analysis and Network Hardening. Situational awareness, as defined earlier, implies knowledge and understanding of both the defender (*knowledge of us*) and the attacker (*knowledge of them*). In turn, this implies knowledge and understanding of all the weaknesses existing in the network we aim to defend. Each host's susceptibility to attack depends on the vulnerabilities of other hosts in the network, as attackers can combine vulnerabilities in unexpected ways, allowing them to incrementally penetrate a network and compromise critical systems. Therefore, to protect critical networks, we must understand not only individual system vulnerabilities, but also their interdependencies. While we cannot predict the origin and timing of attacks, we can reduce their impact by identifying all possible attack paths through our networks. To this aim, we cannot rely on manual processes and mental models. Instead, we need automated tools to analyze and visualize vulnerability dependencies and attack paths, so as to understand the overall security posture of our systems, and provide context over the full security life cycle.

A viable approach to such full-context security is topological vulnerability analysis (TVA) [13]. TVA monitors the state of network assets, maintains models of network vulnerabilities and residual risk, and combines these to produce models that convey the impact of individual and combined vulnerabilities on the overall security posture. The core element of this tool is an attack graph showing all possible ways an attacker can penetrate the network. Topological vulnerability analysis looks at vulnerabilities and their hardening measures within the context of overall network security by modeling their interdependencies via attack graphs. This approach provides a unique new capability, transforming raw security data into a roadmap that lets one proactively prepare for attacks, manage vulnerability risks, and have real-time situational awareness. It supports both offensive (e.g., penetration testing) and defensive (e.g., network hardening) applications. The mapping of attack paths through a network provides a concrete understanding of how individual and combined vulnerabilities impact overall network security. For example, we can (i) determine whether risk-mitigating efforts have a significant impact on overall security; (ii) determine how much a new vulnerability will impact overall security; and (iii) analyze how changes to individual hosts may increase overall risk to the enterprise. This approach has been implemented as a security tool – CAULDRON [17] – which transforms raw security data into an attack graph.

Attack graph analysis can be extended to automatically generate recommendations for hardening networks. *Network hardening* consists in changing network

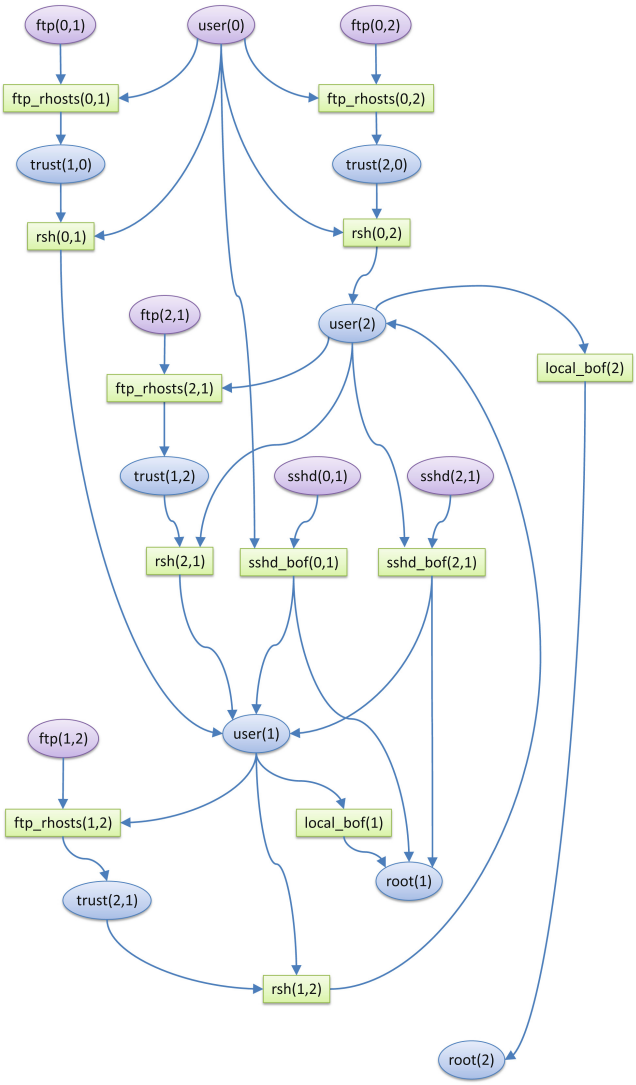

Fig. 9. An example of attack graph (Color figure online)

configurations in such a way to make networks resilient to certain attacks and prevent attackers from reaching certain goals, as shown in the following example.

Figure 9 shows the attack graph for a network of three hosts (referred to as host 0, 1, and 2 respectively), where rectangles represent vulnerabilities and ovals represent security conditions that are either required to exploit a vulnerability (*pre-conditions*) or created as the result of an exploit (*post-conditions*). Purple ovals represent initial conditions – which depend on the initial configuration of the system – whereas blue ovals represent intermediate conditions created as

the result of an exploit. Conceptually, the formalism used in the attack graph of Fig. 9 is equivalent to the formalisms used in Fig. 5 and Fig. 8, but in this case we are explicitly showing the pre- and post-conditions of each vulnerability. In this example, the attacker's objective is to gain administrative privileges on host 2, a condition that is denoted as root(2). In practice, to prevent the attacker from reaching a given security condition, the defender has to prevent the exploitation of each vulnerability that has the target condition as a post-condition. For instance, in the example of Fig. 9, one could prevent the attacker from gaining user privileges on host 1, denoted as user(1), by preventing exploitation of rsh(0,1), rsh(2,1), sshd_bof(0,1), and sshd_bof(2,1). Conversely, to prevent exploitation of a vulnerability, at least one pre-condition must be disabled. For instance, in the example of Fig. 9, one could prevent the attacker from exploiting rsh(1,2) by disabling either trust(2,1) or user(1).

The analysis of attack graphs provides alternative sets of hardening measures that guarantee security of critical systems. For instance, in the example of Fig. 9, one could prevent the attacker from reaching the target security condition root(2) by disabling one of the following two sets of initial conditions: {ftp(0,2), ftp(1,2)}, or {ftp(0,2), ftp(0,1), sshd(0,1)}. Through this unique new capability, administrators are able to determine the best sets of hardening measures that should be applied in their environment. Each set of hardening measures may have a different cost, and administrators can choose hardening solutions that are optimal with respect to a predefined notion of cost. Such hardening solutions prevent the attack from succeeding, while minimizing the associated costs, but, unfortunately, the search space grows exponentially with the size of the attack graph. In applying network hardening to realistic network environments, it is crucial that the algorithms are able to scale. Progress has been made in reducing the complexity of attack graph manipulation so that it scales quadratically – or linearly within defined security zones [23]. However, many approaches for generating hardening recommendations search for exact solutions [26], which is an intractable problem. Another limitation of most work in this area is the assumption that network conditions are hardened independently. This assumption does not hold true in real network environments. Realistically, network administrators can take actions that affect vulnerabilities across the network, such as pushing patches out to many systems at once. Furthermore, the same hardening results may be obtained through more than one action.

Overall, to provide realistic recommendations, the hardening strategy we proposed in [5] takes such factors into account, and removes the assumption of independent hardening actions. We defined a network hardening strategy as a set of allowable atomic actions that administrators can take (e.g., shutting down an ftp server, blacklisting certain IP addresses), each resulting in the removal of multiple initial conditions. A formal cost model was introduced to account for the impact of these hardening actions, which have a cost both in terms of implementation and in terms of loss of availability (e.g., when hardening requires shutting down a vulnerable service). As computing the minimum-cost hardening solution is intractable, we introduced an approximation algorithm that

finds near-optimal solutions while scaling almost linearly – for certain values of the parameters – with the size of the attack graph. Formal analysis shows that a theoretical upper bound exists for the worst-case approximation ratio, whereas experimental results show that, in practice, the approximation ratio is significantly lower than such bound.

Still, we must understand that not all attacks can be prevented, and there might be residual vulnerabilities even after reasonable hardening measures have been applied. We then rely on intrusion detection techniques to identify actual attack instances. But the detection process needs to be tied to residual vulnerabilities, especially the ones that lie on paths to critical network resources as discovered by TVA. Tools such as Snort can analyze network traffic and identify attempts to exploit unpatched vulnerabilities in real time, thus enabling timely response and mitigation efforts. Once attacks are detected, comprehensive capabilities are needed to react to them. TVA can reduce the impact of attacks by providing knowledge of the possible vulnerability paths through the network. Attack graphs can be used to correlate and aggregate network attack events, across platforms as well as across the network. These attack graphs also provide the necessary context for optimal response to ongoing attacks.

In conclusion, topological analysis of vulnerabilities plays an important role in gaining situational awareness, and more specifically what we earlier defined *knowledge of us*. Without automated tools such as CAULDRON, human analysts would be required to manually perform vulnerability analysis, and this would be an extremely tedious and error-prone task. From the example of Fig. 9, it is clear that even a relatively small network may result in a large and complex attack graph. With the introduction of automated tools such as CAULDRON, the role of the analyst shifts towards higher-level tasks: instead of trying to analyze and correlate individual vulnerabilities, analysts are presented with a clear picture of existing vulnerability paths. Instead of trying to manually map alerts to possible vulnerability exploits, analysts are required to validate the findings of the tool and drill down as needed [6]. The revised role of human analysts – while not changing their ultimate mandate and responsibilities – will require them to be properly trained to use and benefit from the new automated tools. Most likely, as their productivity is expected to increase as a result of automating the most repetitive and time-consuming tasks, fewer analysts will be required to monitor a given infrastructure.

2.3 Zero-Day Analysis

As stated earlier, attackers can leverage complex interdependencies among network configurations and vulnerabilities to penetrate seemingly well-guarded networks. Besides well-known weaknesses, attackers may leverage unknown (zero-day) vulnerabilities, which not even developers and administrators are aware of. While attack graphs can reveal potential paths that attackers can take to penetrate networks, they can only provide qualitative results, unless they are augmented with quantitative information, as we did by defining the notion of probabilistic temporal attack graph. However, traditional efforts on network

security metrics typically assign numeric scores to vulnerabilities as their relative exploitability or likelihood, based on known facts about each vulnerability, but this approach is clearly not applicable to zero-day vulnerabilities due to the lack of prior knowledge or experience. In fact, a major criticism of existing efforts on security metrics is that zero-day vulnerabilities are unmeasurable due to the less predictable nature of both the process of introducing software flaws and that of discovering and exploiting vulnerabilities [21]. Relatively recent work addresses the above limitations by proposing a security metric for zero-day vulnerabilities, namely, the k-zero day safety metric [25]. Intuitively, this metric estimates the number k of distinct zero-day vulnerabilities that are needed to compromise a given network asset. A larger value of this metric indicates that the system is relatively more secure against zero-day attacks, because it is less likely that a larger number of different unknown vulnerabilities will all be available at the same time and exploitable by the same attacker. However, as shown in [25], the problem of computing the exact value of k is intractable, and the original approach to estimating the value of k relied on unrealistic assumptions about the availability of a complete zero-day attack graph, which in practice is infeasible for large networks [23].

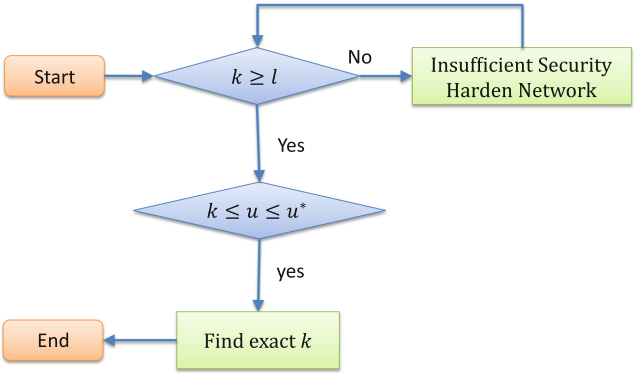

Fig. 10. Flowchart of the zero-day analysis process

In order to address the limitations of previous approaches, we proposed a suite of efficient solutions [7] to enable zero-day analysis of practical applicability to networks of realistic sizes. This approach – which combines on-demand attack graph generation with the evaluation of the k-zero-day safety metric – starts from the problem of deciding whether a given network asset is at least k-zero-day safe for a given value of k (i.e., $k \geq l$), meaning that it satisfies some baseline security requirements: in other words, in order to penetrate a system, an attacker must be able to exploit at least a relatively high number of zero-day vulnerabilities. Second, it identifies an upper bound on the value of k, intuitively corresponding to the maximum security level that can be achieved with respect to this metric.

Finally, if k is large enough, we can assume that the system is sufficiently secure with respect to zero-day attacks. Otherwise, we can compute the exact value of k by efficiently reusing the partial attack graph computed in previous steps (Fig. 10).

In conclusion, similarly to what we discussed at the end of the previous section, the capability presented in this section is critical to gain situation awareness, and can be achieved either manually or automatically. However, given the uncertain nature of zero-day vulnerabilities, the results of manual analysis could be more prone to subjective interpretation than any other capability we discuss in this chapter. At the same time, since automated analysis relies on assumptions about the existence of zero-day vulnerabilities, complete reliance on automated tools may not be the best option for this capability, and a human-in-the-loop solution may provide the most benefits. In fact, the solution presented in [7] can be seen as a decision support system where human analysts can play a role in the overall workflow.

3 Adaptive Cyber Defense

The computer systems, software applications, and network technologies that we use today were developed in user and operator contexts that greatly valued standardization, predictability, and availability. Performance and cost-effectiveness were the main market drivers. It is only relatively recently that security and resilience – not to be confused with fault tolerance – have become equally desirable properties of cyber systems. As a result, the first generation of cyber security technologies largely relied on system hardening through improved software security engineering – to reduce vulnerabilities and attack surfaces – and layering security through defense-in-depth. These security technologies sought to ensure the homogeneity, standardization, and predictability that have been so valued by the market. Consequently, most of our cyber defenses are static. They are governed by slow and deliberative processes such as testing, episodic penetration exercises, security patch deployment, and human-in-the-loop monitoring of security events.

Adversaries benefit greatly from this situation because they can continuously and systematically probe targeted networks with the confidence that those networks will change slowly if at all. Adversaries can afford the time to engineer reliable exploits and plan their attacks in advance. Moreover, once an attack succeeds, adversaries persist for an extended period of time inside compromised networks and hosts, because the hosts, networks, and services – largely designed for availability and homogeneity – do not reconfigure, adapt or regenerate except in deterministic ways to support maintenance and uptime requirements.

To address the limitations of today's approach to cyber defense, researchers have recently started to investigate various approaches – collectively referred to as Adaptation Techniques (AT) – to make networked information systems less homogeneous and less predictable. We provide an overview of adaptation techniques in Sect. 3.1, whereas in Sect. 3.2 we briefly describe a framework we

proposed to address the problem of quantifying the effectiveness and cost of different adaptive techniques.

3.1 Adaptation Techniques

The basic idea of Adaptation Techniques (AT) is to engineer systems that have homogeneous functionality but randomized manifestations. Homogeneous functionality allows authorized use of networks and services in predictable, standardized ways, whereas randomized manifestations make it difficult for attackers to engineer exploits remotely, let alone parlay one exploit into successful attacks against a multiplicity of hosts. Ideally, each compromise would require the same, significant effort by the attacker.

In general, with the term *adaptation techniques*, we refer to concepts such as Moving Target Defense (MTD) [14,15] as well as artificial diversity and bio-inspired defenses to the extent that they involve system adaption for security and resiliency purposes. In the following, we will use the terms *adaptation technique* and *ACD technique* interchangeably.

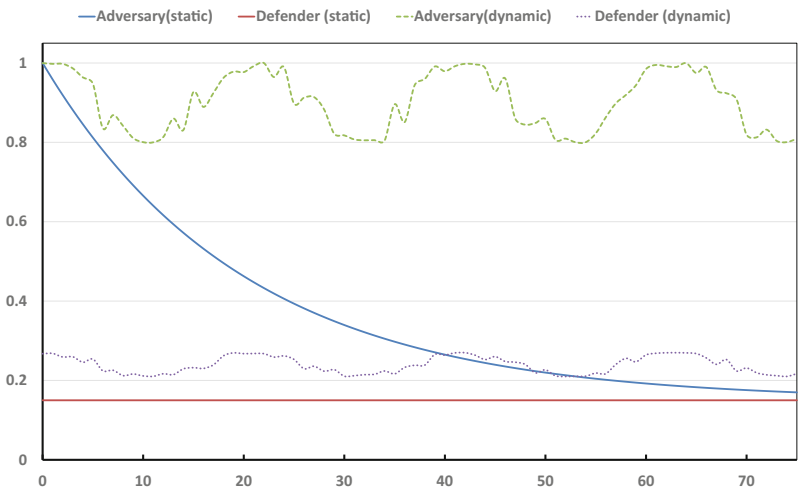

Fig. 11. Adversary vs. defender uncertainty before and after deployment of ACD techiniques

ACD techniques increase complexity and cost for the attackers by continuously changing or shifting a system's *attack surface*, which has been defined as the *"subset of the system's resources (methods, channels, and data) that can be potentially used by an attacker to launch an attack"* [20]. Thus, the majority of ACD techniques operate by periodically reconfiguring one or more system parameters in order to offer randomized manifestations of the system and disrupt any knowledge an attacker may have acquired. Different ACDs may be

designed to address different stages of the Cyber Kill Chain, a framework developed by Lockheed Martin as part of the Intelligence Driven Defense model for identification and prevention of cyber intrusions activity [12]. The majority of the techniques currently available are designed to address the reconnaissance phase of the cyber kill chain, as they attempt to interfere with the attacker's effort to gather information about the target system.

One of the major drawbacks of many ACDs is that they force the defender to periodically reconfigure the system, which may introduce a costly overhead to legitimate users, as well as the potential for denial of service. Additionally, most existing techniques are purely proactive in nature or do not adequately consider the attacker's behavior. To address this limitation, alternative approaches aim at inducing a "perceived" attack surface by deceiving the attacker into making incorrect inferences about the system's configuration [3], rather than actually reconfiguring the system. Honeypots have also been used to divert attackers away from critical resources [1], but they have proven to be less effective than ACDs because they provide a static solution: once a honeypot has been discovered, the attacker will simply avoid it. One of the primary goals of dynamically changing the attack surface of a system is to increase the uncertainty for the adversary, while limiting the overhead for the defender. The notional diagram in Fig. 11 shows how the level of uncertainty about network topology and configuration may vary over time for both the attacker and the defender, before and after the deployment of adaptation techniques. In a static configuration (i.e., before deploying any adaptation technique), adversaries can improve their knowledge of the target system over time, thus reducing their uncertainty. At the same time the defender's uncertainty remains a constant low level.

When ACD mechanisms are deployed, each reconfiguration of the system invalidates some of the information previously acquired by the attacker, thus increasing the adversary's uncertainty. Before the attack surface is changed again, the adversary will be able to regain some knowledge and temporarily reduce the uncertainty, but this effort will be again defeated with the next reconfiguration. Figure 11 shows that the adversary's uncertainty would in fact fluctuate, but will always remain above a certain relatively high threshold. We also need to consider that any of the proposed adaptation mechanisms introduces uncertainty for the defender as well, albeit less than that introduced for the adversary. As long as attack surface reconfiguration mechanisms include a secure protocol for informing all legitimate entities about the changes, the defender's uncertainty can be contained within manageable levels, and the defender can maintain an advantage over the adversary. Figure 11 shows that, before deploying any ACD mechanism, the uncertainty gap between defender and adversary decreases over time, thus eroding the defender's advantage. On the other hand, when the attack surface is dynamically changed, the uncertainty gap remains consistently high over time.

Examples of adaptation techniques include randomized network addressing and layout, obfuscated OS types and services, randomized instruction set and memory layout, randomized compiling, just-in-time compiling and decryption,

dynamic virtualization, workload and service migration, and system regeneration, to name a few. Each of these techniques has a performance and maintenance cost associated with it. For example, randomized instruction set and memory layout clearly limit the extent to which a single buffer overflow exploit can be used to compromise a collection of hosts. However, it also makes it more difficult for system administrators and software vendors to debug and update hosts because all the binaries are different. Furthermore, randomized instruction set and memory layout techniques will not make it more difficult for an attacker to determine a network's layout and its available services. Similar analyses are possible for each of the techniques listed above. For example, randomizing network addresses makes it more difficult for an adversary to perform reconnaissance on a target network remotely, but does not make it more difficult for the attacker to exploit a specific host once it is identified and reachable.

While a variety of different ACD techniques exist, the contexts in which they are useful and their added cost to the defenders (in terms of performance and maintainability) can vary significantly. In fact, the majority of ACD research has been focused on developing specific new techniques as opposed to understanding their overall operational costs, when they are most useful, and what their possible inter-relationships might be. In fact, while each ACD approach might have some engineering rigor, the overall discipline is largely ad hoc when it comes to understanding the totality of ACD methods and their optimized application.

3.2 Quantification Framework

In this section, we discuss the quantification framework we proposed in [10] to address current limitations of ACD research with respect to quantification, and to enable comparative analysis of different techniques. The framework was specifically developed for quantification of moving target defense techniques, but it can be easily generalized to address the broader scope of ACD techniques.

The model, as shown for the example in Fig. 12, consists of four layers: (i) a service layer representing the set $\mathcal{S}$ of services to be protected; (ii) a weakness layer representing the set $\mathcal{W}$ of general classes of weaknesses that may be exploited; (iii) a knowledge layer representing the set $\mathcal{K}$ of all possible knowledge blocks required to exploit those weaknesses; and (iv) an MTD layer representing the set $\mathcal{M}$ of available MTD techniques. In the simple example of Fig. 12, (i) the service to be protected is a database server; (ii) the two classes of weaknesses that could be exploited are represented by vulnerabilities enabling SQL injection and buffer overflow respectively; (iii) the knowledge blocks needed to exploit such vulnerabilities include knowledge of the service, its IP address, and memory layout; and (iv) three MTD techniques are available to protect such knowledge, namely, Service Rotation, IP Rotation, and Address Space Layout Randomization (ASLR).

The proposed MTD quantification framework can be formally defined as a 7-tuple $(\mathcal{S}, \mathcal{R}_{SW}, \mathcal{W}, \mathcal{R}_{WK}, \mathcal{K}, \mathcal{R}_{KM}, \mathcal{M})$, where: (i) $\mathcal{S}$, $\mathcal{W}$, $\mathcal{K}$, $\mathcal{M}$ are the sets of services, weaknesses, knowledge blocks, and MTD techniques, respectively; (ii) $\mathcal{R}_{SW} \subseteq \mathcal{S} \times \mathcal{W}$ represents relationships between services and the common

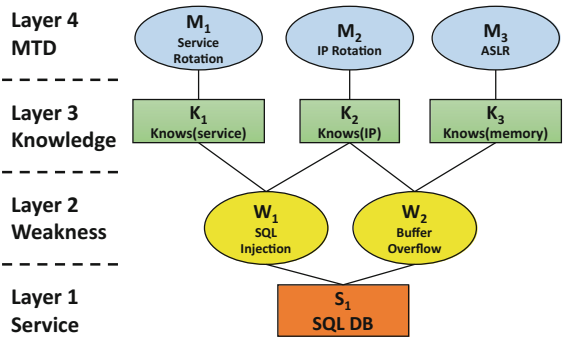

Fig. 12. Layers of the quantification model

weaknesses they are vulnerable to; (iii) $\mathcal{R}_{WK} \subseteq \mathcal{W} \times \mathcal{K}$ represents relationships between weaknesses and the knowledge blocks required to exploit them; and (iv) $\mathcal{R}_{KM} \subseteq \mathcal{K} \times \mathcal{M}$ represents relationships between knowledge blocks and the MTD techniques that can protect them. The proposed model induces a k-partite graph (with $k = 4$) $G = (\mathcal{S} \cup \mathcal{W} \cup \mathcal{K} \cup \mathcal{M}, \mathcal{R}_{SW} \cup \mathcal{R}_{WK} \cup \mathcal{R}_{KM})$. The four layers of the model are discussed in more details in the following subsections.

Layer 1: Service Layer. The first layer represents the set $\mathcal{S}$ of services we wish to protect against attacks. We assume that the services are time-invariant, i.e., the functionality of the services does not change over time, and services cannot be taken down to prevent attacks, as this action would result in a denial of service to legitimate users. In the example of Fig. 12, for the sake of presentation, we considered only one service, but the model can be easily extended to consider multiple interdependent services that may be exploited and compromised in a multi-step attack, similarly to how exploit chains within attack graphs might be exploited by an attacker [18,24].

Layer 2: Weakness Layer. The second layer represents the set of weaknesses $\mathcal{W}$ that services are vulnerable to. We choose general classes of weaknesses, rather than specific vulnerabilities, because there are too many vulnerabilities to enumerate, some vulnerabilities are unknown, and, depending on the MTD used (e.g., OS rotation), specific vulnerabilities may change over time. Using general classes of weaknesses when building the model makes it time-invariant. The classes of weaknesses used in our model are draw primarily from MITRE's Common Weakness Enumeration (CWE) project [9], particularly from those known as the *"Top 25 Most Dangerous Software Errors."* Although many of the top software errors are primarily the result of bad coding practices and better solved at development time, the top software errors enabling exploits such as *SQL Injection*, *OS Injection*, and *Classic Buffer Overflow* can be addressed at runtime by MTDs (e.g., SQLrand [8]) and make for good general categories of weaknesses.

Layer 3: Knowledge Layer. The third layer represents the knowledge blocks $\mathcal{K}$ required to exploit weaknesses in $\mathcal{W}$. We assume that knowledge blocks are independent and must be acquired using different methods. For example, IP address and port number of a target service should not modeled as separate knowledge blocks because a method to determine one would also reveal the other.

The relationship between the knowledge and weakness layers is many-to-many. A weakness may require several pieces of knowledge to be exploited, and a knowledge block may be key to exploiting several weaknesses. This layer may also be extended as new MTDs – disrupting new and different aspects of the attacker's knowledge – are developed.

In our example, we assume that, in order to execute a SQL injection attack, an attacker must gather information about the service (e.g., name and version of the specific DBMS) and network configuration (e.g., IP address). In order to execute a buffer overflow attack, an attacker must know the IP address and some information about the vulnerable memory locations. A higher-fidelity version of this model may take a knowledge block and break it down into finer-grained items that are specifically targeted by available MTDs.

Layer 4: MTD Layer. The fourth layer of the model represents the set $\mathcal{M}$ of available MTDs. As MTD techniques provide probabilistic security, we model the impact of an MTD M_i on the attacker's effort to acquire knowledge K_j by associating a probability $P_{i,j}$ – representing the attacker's success rate – with the relation (K_j, M_i). As mentioned earlier, when only static defenses are deployed, an attacker will acquire the necessary knowledge without significant effort, which we model by associating a probability of 1. For example, if technique M_1 in Fig. 12 (*Service Rotation*) reduces an attacker's likelihood of acquiring knowledge block K_1 (i.e., correct version of the service) by 60%, we would label that edge with $P_{1,1} = 0.4$. The exact methodology for determining the value of $P_{i,j}$ may depend on the specific nature of individual MTDs, however, expressing MTD effectiveness in terms of the probability that an attacker will succeed in acquiring required knowledge enables us to evaluate multiple different techniques using a uniform approach.

4 Conclusions and Future Work

In this chapter, we started from the observation that today's cyber security landscape is asymmetric and tends to favor the attacker over the defender. We then discussed the challenging problem of reducing the attacker's advantage, and potentially leveling the cyber playing field. We showed that, in order to achieve this goal, one possible solution is to *attack* the problem on two fronts. On one side, to reduce the defender's effort, we can improve the defender's understanding of multiple aspects of the cyber landscape through Cyber Situational Awareness techniques. On the other side, to increase the attacker's effort, we can introduce uncertainty about information on the target system through Adaptive Cyber

Defense techniques. We presented an overview of these two research areas, and discussed some representative contributions within each of them.

Current research in these relatively new areas has clearly shown promise to significantly enhance our defensive capabilities. However, much work remains to be done if we want to push our CSA and ACD capabilities beyond simply leveling the cyber playing field. Ideally, we would like to completely reverse the intrinsic asymmetry of today's cyber security landscape in favor of the defender. To achieve this goal, several research directions will need to be further investigated, including adversarial modeling, game and control theoretic approaches to security, artificial intelligence techniques, and human-computer interfaces. We envision a future where human analysts will work side-by-side with automated tools, thus requiring more sophisticated human-computer interaction mechanisms and protocols. Such a closer interaction will help form better situational awareness in a timely and cost-effective manner, and will enable defenders to proactively prepare to face anticipated threats and to quickly adapt to an ever-evolving cyber landscape.

Acknowledgement. This work was partially supported by the Army Research Office under grants W911NF-09-1-0525 and W911NF-13-1-0421.

References

1. Abbasi, F.H., Harris, R.J., Moretti, G., Haider, A., Anwar, N.: Classification of malicious network streams using honeynets. In: Proceedings of the IEEE Global Communications Conference (IEEE GLOBECOM 2012), pp. 891–897. IEEE, Anaheim, CA, USA, December 2012
2. Albanese, M., Jajodia, S.: Formation of awareness. In: Kott, A., Wang, C., Erbacher, R.F. (eds.) Cyber Defense and Situational Awareness. AIS, vol. 62, pp. 47–62. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11391-3_4
3. Albanese, M., Battista, E., Jajodia, S., Casola, V.: Manipulating the attacker's view of a system's attack surface. In: IEEE Conference on Communications and Network Security, CNS 2014, pp. 472–480, San Francisco, CA, USA, October 2014
4. Albanese, M., Jajodia, S.: A graphical model to assess the impact of multi-step attacks. J. Def. Model. Simul. **15**(1), 79–93 (2018)
5. Albanese, M., Jajodia, S., Noel, S.: Time-efficient and cost-effective network hardening using attack graphs. In: Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), Boston, MA, USA, June 2012
6. Albanese, M., Jajodia, S., Pugliese, A., Subrahmanian, V.S.: Scalable analysis of attack scenarios. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 416–433. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23822-2_23
7. Albanese, M., Jajodia, S., Singhal, A., Wang, L.: An efficient approach to assessing the risk of zero-day. In: Samarati, P. (ed.) Proceedings of the 10th International Conference on Security and Cryptography (SECRYPT 2013), pp. 207–218. SciTePress, Reykjavík, Iceland (July 2013)

8. Boyd, S.W., Keromytis, A.D.: SQLrand: preventing SQL injection attacks. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 292–302. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24852-1_21
9. Christey, S.: 2011 CWE/SANS top 25 most dangerous software errors (2011). http://cwe.mitre.org/top25/
10. Connell, W., Albanese, M., Venkatesan, S.: A framework for moving target defense quantification. In: De Capitani di Vimercati, S., Martinelli, F. (eds.) SEC 2017. IAICT, vol. 502, pp. 124–138. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58469-0_9
11. Cybenko, G., Jajodia, S., Wellman, M.P., Liu, P.: Adversarial and uncertain reasoning for adaptive cyber defense: building the scientific foundation. In: Prakash, A., Shyamasundar, R. (eds.) ICISS 2014. LNCS, vol. 8880, pp. 1–8. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13841-1_1
12. Hutchins, E.M., Cloppert, M.J., Amin, R.M.: Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. Lockheed Martin Corporation, Bethesda (2010)
13. Jajodia, S., Noel, S.: Topological vulnerability analysis. In: Jajodia, S., Liu, P., Swarup, V., Wang, C. (eds.) Cyber Situational Awareness. Advances in Information Security, vol. 46, pp. 139–154. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-0140-8_7
14. Jajodia, S., Ghosh, A.K., Subrahmanian, V.S., Swarup, V., Wang, C., Wang, X.S. (eds.): Moving Target Defense II: Application of Game Theory and Adversarial Modeling. Advances in Information Security, vol. 100. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-5416-8
15. Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Wang, X.S. (eds.): Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats. Advances in Information Security, vol. 54. Springer, New York (2011). https://doi.org/10.1007/978-1-4614-0977-9
16. Jajodia, S., Liu, P., Swarup, V., Wang, C. (eds.): Cyber Situational Awareness: Issues and Research. Advances in Information Security. Springer, New York (2010). https://doi.org/10.1007/978-1-4419-0140-8
17. Jajodia, S., Noel, S., Kalapa, P., Albanese, M., Williams, J.: Cauldron: mission-centric cyber situational awareness with defense in depth. In: Proceedings of the Military Communications Conference (MILCOM 2011), pp. 1339–1344. Baltimore, MD, USA, November 2011
18. Jajodia, S., Noel, S., O'Berry, B.: Topological analysis of network attack vulnerability. In: Kumar, V., Srivastava, J., Lazarevic, A. (eds.) Managing Cyber Threats: Issues, Approaches, and Challenges. MACO, vol. 5, pp. 247–266. Springer, Boston (2005). https://doi.org/10.1007/0-387-24230-9_9
19. Leversage, D.J., Byres, E.J.: Estimating a system's mean time-to-compromise. IEEE Secur. Priv. **6**(1), 52–60 (2008)
20. Manadhata, P.K., Wing, J.M.: An attack surface metric. IEEE Trans. Software Eng. **37**(3), 371–386 (2011)
21. McHugh, J.: Quality of protection: measuring the unmeasurable? In: Proceedings of the 2nd ACM Workshop on Quality of Protection (QoP 2006), pp. 1–2. ACM, Alexandria, VA, USA, October 2006
22. Natrajan, A., Ning, P., Liu, Y., Jajodia, S., Hutchinson, S.E.: NSDMiner: Automated discovery of network service dependencies. In: Proceedings of the 31st Annual International Conference on Computer Communications (INFOCOM 2012), pp. 2507–2515, Orlando, FL, USA, March 2012

23. Noel, S., Jajodia, S.: Managing attack graph complexity through visual hierarchical aggregation. In: Proceedings of the ACM CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC 2004), pp. 109–118. ACM, Fairfax, VA, USA, October 2004
24. Wang, L., Islam, T., Long, T., Singhal, A., Jajodia, S.: An attack graph-based probabilistic security metric. In: Atluri, V. (ed.) DBSec 2008. LNCS, vol. 5094, pp. 283–296. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70567-3_22
25. Wang, L., Jajodia, S., Singhal, A., Noel, S.: k-zero day safety: measuring the security risk of networks against unknown attacks. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 573–587. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15497-3_35
26. Wang, L., Noel, S., Jajodia, S.: Minimum-cost network hardening using attack graphs. Comput. Commun. **29**(18), 3812–3824 (2006)