

An Integrated Framework for Cyber Situation Awareness

Sushil Jajodia and Massimiliano Albanese^(✉)

Center for Secure Information Systems, George Mason University,
Fairfax, VA 22030, USA
{jajodia,malbanes}@gmu.edu

Abstract. In this chapter, we present a framework that integrates an array of techniques and automated tools designed with the objective of drastically enhancing the Cyber Situation Awareness process. This framework incorporates the theory and the tools we developed to answer – *automatically* and *efficiently* – some of the fundamental questions security analysts may need to ask in the context of Cyber Situation Awareness. Most of the work presented in this chapter is the result of the research effort conducted by the authors as part of a the Multidisciplinary University Research Initiative project sponsored by the Army Research Office that was mentioned in the introductory chapter. We present the key challenges the research community has been called to address in this space, and describe our major accomplishments in tackling those challenges.

1 Introduction

Without loss of generality, the process of situation awareness can be viewed as a three-phase process: situation perception, situation comprehension, and situation projection [1]. *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. *Projection* of the elements of the environment (situation) into the near future encompasses the ability to make predictions based on the knowledge acquired through perception and comprehension.

In order to make informed decisions, security analysts need to be aware of the current situation, the impact and evolution of an attack, the behavior of the attackers, the quality of available information and models, and the plausible futures of the current situation. Some of the questions they may ask are: Is there any ongoing attack? If so, where is the attacker? Are available attack models sufficient to understand what is observed? Can they predict an attacker's goal? If so, how can they prevent that goal from being reached?

This work was supported in part by the Army Research Office MURI awards number W911NF-09-1-0525 and W911NF-13-1-0421 and by the Office of Naval Research grant number N00014-15-1-2007.

In this chapter, we describe several techniques, mechanisms, and tools that can help form and leverage specific types of cyber situation awareness. This framework aims at enhancing the traditional cyber defense process by automating many of the capabilities that have traditionally required a significant involvement of human analysts and other individuals. Ideally, we envision the evolution of the current human-in-the-loop approach to cyber defense into a human-on-the-loop approach, where human analysts would only be responsible for examining and validating or sanitizing the results generated by automated tools, rather than being forced to comb through daunting amounts of log entries and security alerts.

The remainder of this chapter is organized as follows. Section 2 discusses the overall process of Cyber Situation Awareness, and Sect. 3 presents a motivating example we use throughout the chapter. Section 4 introduces the proposed framework, whereas Sect. 5 discusses our scientific progress and major research accomplishments in more detail. Finally, Sect. 6 gives some concluding remarks.

2 The Process of Cyber Situation Awareness

The security analyst – or cyber defense 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 to the organization. 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. Ideally, such tools should be able to automatically answer most, if not all, the questions an analyst may ask about the current situation, the impact and evolution of an attack, the behavior of the attackers, the quality of available information and models, and the plausible futures of the current situation. In the following, we define the fundamental questions that an effective Cyber Situation Awareness framework must be able to help answer. For each question, we identify the inputs as well the outputs of the Cyber Situation Awareness process, and we also briefly comment on the life cycle of the situation awareness gained in response to each question.

1. **Current situation.** *Is there any ongoing attack? If yes, what is the stage of the intrusion and where is the attacker?*

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 is represented by 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 intrusive activities. This type of CSA may quickly become obsolete – if not acted upon timely or updated frequently – as the intruder progresses within the system.

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, so far, 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 intrusive activities. As for the previous case, this type of CSA must be frequently updated to remain useful, because damage will increase as the attack progresses.

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 situation 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 address the *useful life* limitations highlighted above and *refresh* the situation awareness generated in response to the first two sets of questions.

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 its 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 models, 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 situation? What was he trying to achieve?*

Answering this set of questions implies the capability of analyzing the 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 these questions, from the situation 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 set of questions implies the capability of predicting possible moves an attacker may take in the future. With respect to this set of questions, the input to the CSA process is represented by the situation awareness gained in response to the first (or 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 in order to answer all other sets of question the overall CSA process is aiming to address. 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 situation 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 [1]. Situation awareness is generated or used by different mechanisms and tools aimed at addressing the seven classes of questions that security analysts may routinely ask while performing their work tasks.

3 Motivating Example

Throughout this chapter, we will often refer to the network depicted in Fig. 1 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 services, and each of them includes a host accessible from the Internet. The third subnetwork implements the core 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.

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, in order to study the vulnerability landscape of any enterprise network, we extensively use attack graphs, which reveal potential threats by enumerating paths that attackers can take to penetrate a network [8].

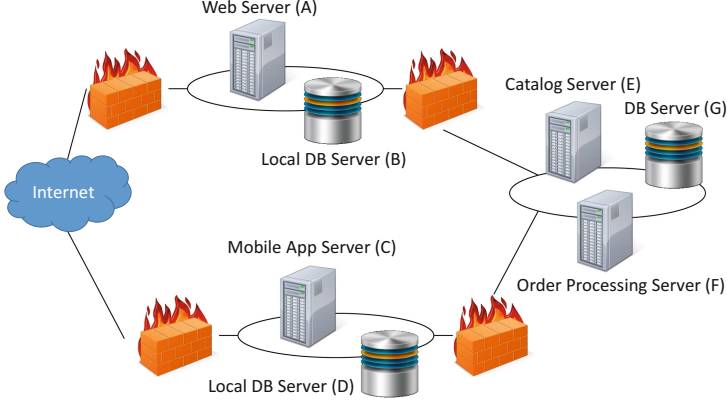


Fig. 1. Motivating example: enterprise network offering two public-facing services

The attack graph for the network of Fig. 1 is shown in Fig. 2. This attack graph 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 pattern 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.

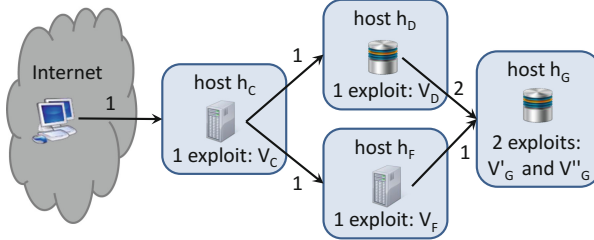


Fig. 2. The attack graph for the motivating example network

4 The Cyber Situation Awareness Framework

The proposed Cyber Situation Awareness framework is illustrated in Fig. 3. We start from analyzing the topology of the network, known vulnerabilities, possible zero-day vulnerabilities – these must be hypothesized – and their interdependencies. Vulnerabilities are often interdependent, making traditional point-wise vulnerability analysis ineffective. Our topological approach to vulnerability analysis

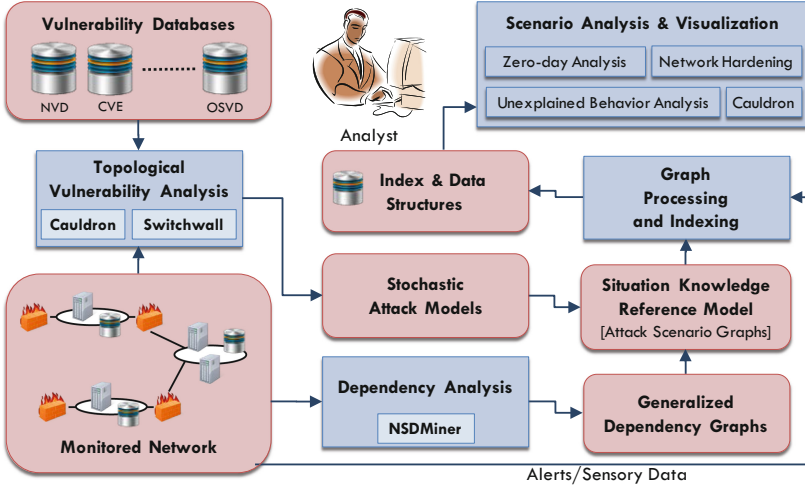


Fig. 3. The proposed Cyber Situation Awareness framework

allows to generate accurate attack graphs showing all the possible attack paths within the network.

A node in an attack graph represents – depending on the level of abstraction – an exploitable vulnerability (or family of exploitable 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/or hosts and derive dependency graphs encoding how these components depend on one another. Dependency analysis is critical to assess current damage caused by ongoing attacks (i.e., the value or utility of services disrupted by the 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.

The dependency graph for the network of Fig. 1 is shown in Fig. 4. 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 then 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 then compute an estimate of the future damage that ongoing attacks might cause for each possible outcome

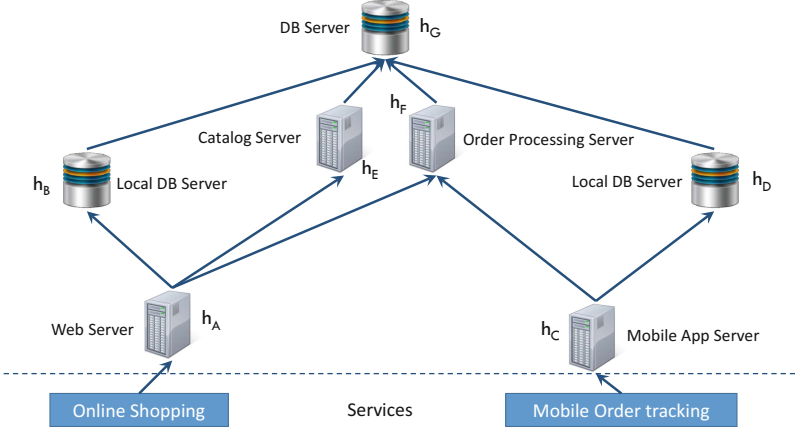


Fig. 4. The dependency graph for the motivating example network

of the current situation. In practice, the proposed attack scenario graph bridges the semantic gap between known vulnerabilities – the lowest abstraction level – and the missions or services that could be ultimately affected by the exploitation of such vulnerabilities – the highest abstraction level. The attack scenario graph for the network of Fig. 1 is shown in Fig. 5. In this figure, the graph on the left is an attack graph modeling all the vulnerabilities in the system and their relationships, whereas the graph on the right is a dependency graph capturing all the explicit and implicit dependencies between services and hosts. 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.

In order to enable concurrent monitoring of multiple attack types, we developed novel graph-based data structures and an index structure to index large amounts of alerts and event data in real-time. We also developed efficient algorithms to analyze such data structures and help automatically answers questions about the current cyber landscape and its evolution.

Moreover, the novel capabilities described so far have been leveraged to develop a suite of additional capabilities and tools, including but not limited to: topological vulnerability analysis [6], network hardening [3], and zero-day analysis [5]. Some of these capabilities and tools are discussed in the following section.

In summary, the proposed framework can provide security analysts with a high-level view of the cyber situation. From the simple example of Fig. 5 – 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 graph of Fig. 5 provides analysts with a visual and very clear understanding of the situation, thus enabling them to focus on higher-level tasks that

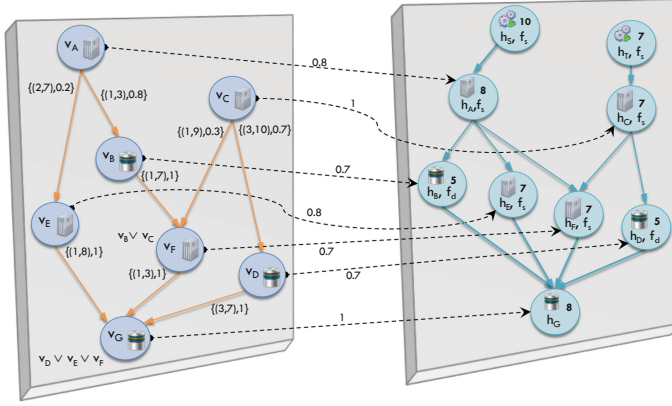


Fig. 5. The attack scenario graph for the motivating example network

require experience and intuition, and thus are more difficult to automate. Additionally, other classes of automated analytical processes may be developed within this framework to support the analyst during these higher-level tasks as well. For instance, based on the model of Fig. 5, we 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 (e.g., sets of network hardening actions).

5 Scientific Progress and Major Accomplishments

In this section, we highlight major accomplishments achieved during the execution of the research project that led to the development of the framework discussed in the previous section.

5.1 Topological Vulnerability Analysis

Situation awareness, as defined in the previous sections, 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 computing infrastructure we aim to defend. By their very nature, security concerns on networks are highly interdependent. Each host's susceptibility to attack depends on the vulnerabilities of other hosts in the network. Attackers can combine vulnerabilities in unexpected ways, allowing them to incrementally penetrate a network and compromise critical systems. To protect our critical infrastructure 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 knowing the possible attack paths through our networks. We need to transform raw security

data into roadmaps that let us proactively prepare for attacks, manage vulnerability risks, and have real-time situation awareness. We cannot rely on manual processes and mental models. Instead, we need automated tools to analyze and visualize vulnerability dependencies and attack paths, so we can understand our overall security posture, providing context over the full security life cycle.

A viable approach to such full-context security is called topological vulnerability analysis (TVA) [6]. 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 protective 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 situation awareness. It supports both offensive (e.g., penetration testing) and defensive (e.g., network hardening) applications. The mapping of attack paths through a network via TVA 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 [7] – which transforms raw security data into a model of all possible network attack paths. In developing this tool, several technical challenges have been addressed, including the design of appropriate models, efficient model population, effective visualization and decision support tools, and the development of scalable mathematical representations and algorithms. The result is a working software tool that offers truly unique capabilities.

Figure 6 shows a simplified attack graph for a network of three hosts (referred to as host 0, 1, and 2 respectively). Rectangles represent vulnerabilities that an attacker may exploit, whereas 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. 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 exploitation of all vulnerabilities that have that condition as a post-condition. For instance, in the example of Fig. 6, 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. 6, one

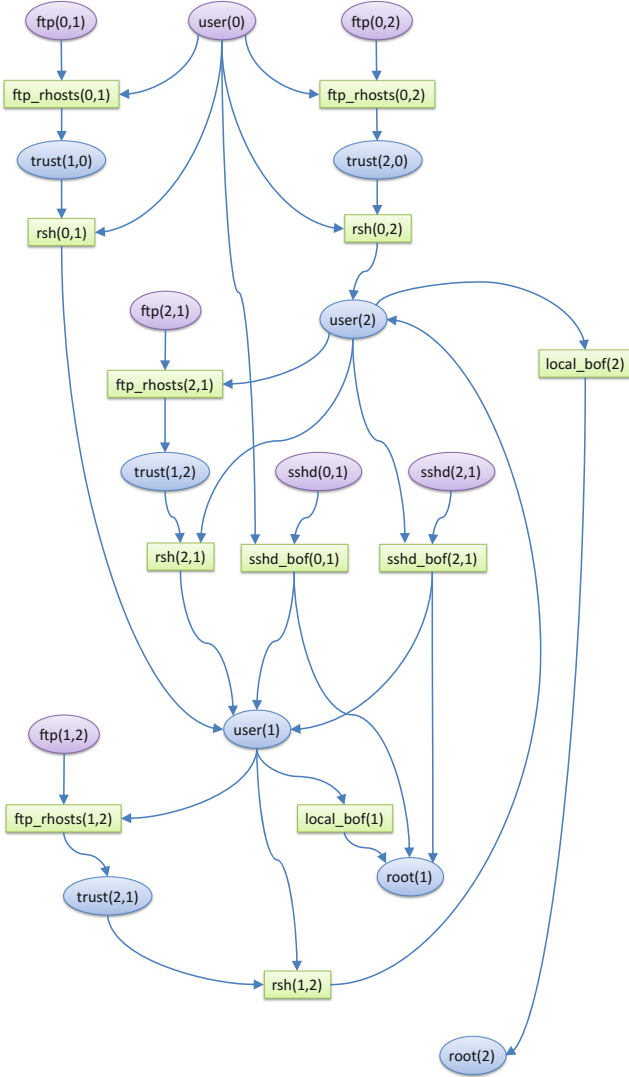


Fig. 6. An example of attack graph (Color figure online)

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 protective measures that guarantee safety of critical systems. For instance, in the example of Fig. 6, 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 protective measures that should be applied in their environment. In fact, each set of protective measures may have a different cost or impact, and administrators can choose the best option with respect to any of these variables.

Still, we must understand that not all attacks can be prevented, and there must usually remain some residual vulnerability even after reasonable protective 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 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. TVA 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 situation 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. 6, 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 have in front of them 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 [4]. The revised role of human analysts – while not changing their ultimate mandate and responsibilities – will require that they are 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.

5.2 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 even developers are not aware of. In-depth analysis of network vulnerabilities must consider attacker exploits not merely in isolation, but in combination. Attack graphs reveal such threats by enumerating potential paths that attackers can take to penetrate networks. This helps determine whether a given set of network hardening measures provides safety of given critical assets. However, attack graphs can only provide qualitative results (i.e.,

secure or insecure), and this may render resulting hardening recommendations ineffective or far from optimal.

To address these limitations, 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. However, 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 [10]. Recent work addresses the above limitations by proposing a security metric for zero-day vulnerabilities, namely, k -zero day safety [14]. Intuitively, this metric is based on the number of distinct zero-day vulnerabilities that are needed to compromise a given network asset. A larger such number indicates relatively more security, because it is less likely to have a larger number of different unknown vulnerabilities all available at the same time, applicable to the same network, and exploitable by the same attacker. However, as shown in [14], the problem of computing the exact value of k is intractable. Moreover, Wang et al. [14] assume the existence of a complete attack graph, but, unfortunately, generating zero-day attack graphs for large networks is usually infeasible in practice [13]. These facts comprise a major limitation in applying this metric or any other similar metric based on attack graphs.

In order to address the limitations of existing approaches, we proposed a set of efficient solutions [5] 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 k -zero-day safety – starts from the problem of deciding whether a given network asset is at least k -zero-day safe for a given value of k , 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.

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 [5] can

be seen as a decision support system where human analysts can play a role in the overall workflow.

5.3 Network Hardening

As discussed earlier, attack graphs reveal threats by enumerating potential paths that attackers can take to penetrate networks. Attack graph analysis can be extended to automatically generate recommendations for hardening networks, which consists in changing network configurations in such a way to make networks resilient to certain attacks and prevent attackers from reaching certain goals. One must consider combinations of network conditions to harden, which has corresponding impact on removing paths in the attack graph. For instance, in Sect. 5.1, we discussed how one could prevent the attacker from reaching the target security condition $\text{root}(2)$ in the example of Fig. 6, and we identified two possible hardening solutions. Furthermore, one can generate hardening solutions that are optimal with respect to some notion of cost. Such hardening solutions prevent the attack from succeeding, while minimizing the associated costs. However, the general solution to optimal network hardening scales exponentially as the number of hardening options itself scales 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 [13]. However, many approaches for generating hardening recommendations search for exact solutions [15], 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 [3] takes such factors into account, and removes the assumption of independent hardening actions. We define 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) and that involve hardening multiple network conditions. A formal cost model is introduced to account for the impact of these hardening actions. Each hardening action has a cost both in terms of implementation and in terms of loss of productivity (e.g., when hardening requires shutting down a vulnerable service). This model allows the definition of hardening costs that accurately reflect realistic network environments. Because computing the minimum-cost hardening solution is intractable, we introduce an approximation algorithm to compute suboptimal hardening solutions. This algorithm finds near-optimal solutions while scaling almost linearly – for certain values of the parameters – with the size of the attack graph, as confirmed by experimental evaluations. Finally, theoretical analysis shows that there is a

theoretical upper bound for the worst-case approximation ratio, whereas experimental results show that, in practice, the approximation ratio is much lower than such bound, that is, the solutions found using this approach are not far, in terms of cost, from the optimal solution. In conclusion, automated analysis of network hardening options can greatly improve the performance of a security analyst, by providing a timely list of recommended strategies to prevent attackers from compromising the target system while, at the same time, minimizing the cost for the defender. The analyst will then be responsible solely for validating the recommended strategies and selecting the ones that appear to be the most effective in meeting not only quantitative but also qualitative requirements. For instance, automated analysis may conclude that the most cost-effective hardening solution is one that requires – amongst other things – to temporarily shut down the server hosting the company’s web site. Although the website may not be running any revenue-generating services, the potential impact on the company’s reputation may make this solution less attractive, and a human analyst looking at the results of the automated tools may opt for the second-best solution after taking into account factors that the tools were not able to capture.

5.4 Probabilistic Temporal Attack Graph

The first step in achieving any level of automation in the situation awareness process is to develop the capability of modeling cyber-attacks and their consequences. This capability is critical to support many of the additional capabilities needed to address the key questions presented earlier in this chapter (e.g., modeling the attacker, predicting future scenarios).

Attack graphs have been widely used to model attack patterns, and to correlate alerts. However, existing approaches typically do not provide mechanisms for evaluating the likelihood of each attack pattern or its impact on the organization or mission. To address this limitation, we extend the attack graph model discussed earlier in this chapter with the notion of *timespan distribution*, which encodes probabilistic knowledge of the attacker’s behavior as well as temporal constraints on the unfolding of attacks. We assume that each step of an attack sequence is completed within certain temporal windows after the previous exploit has been executed, each associated with a probability. For instance, suppose an attacker has gained some privileges on host h_E in Fig. 1. Using these privileges, he can then create the conditions to exploit a vulnerability on the main database server. However, this will take a variable amount of time depending on his skill level. The attacker will then have time to exploit the vulnerability until the vulnerability itself is patched, or the attack is discovered.

Leverage and Byres [9] describe how to estimate the *mean time to compromise* a system and relate that to the skill level of the attacker. This approach can be generalized to estimate timespan distributions for individual vulnerability exploits. In fact, we can assume that the time taken to exploit a vulnerability varies with the skill level of the attacker. Additionally, some vulnerabilities are easier to exploit than others, thus exhibiting higher probabilities. Intuitively, a timespan distribution specifies a set of disjoint time intervals when a given

exploit might be executed, and an incomplete probability distribution over such time intervals.

In our model, edges in the attack graph are labeled with timespan distributions. For instance, in the attack graph of Fig. 5, the edges from V_A to V_E and V_B are labeled with $\{(2, 7), 0.2\}$ and $\{(1, 3), 0.8\}$ respectively, meaning that, after exploiting V_A , with 20% probability an attacker will exploit V_E between 2 and 7 time units later, and with 80% probability he will exploit V_B between 1 and 3 time units later.

5.5 Dependency Graph

Government or enterprise networks today host a wide variety of network services, which often depend on one another to provide and support network-based services and applications. Understanding such dependencies is essential for maintaining the well-being of a network and its applications, particularly in the presence of network attacks and failures. In a typical government or enterprise network, which is complex and dynamic in configuration, it is non-trivial to identify all these services and their dependencies. Several techniques have been developed to learn such dependencies automatically. However, they are either too complex to fine-tune or cluttered with false positives and/or false negatives.

We developed several novel techniques as well as a tool named NSDMiner (which stands for Network Service Dependencies Miner) to automatically discover the dependencies between network services from passively collected network traffic [11]. NSDMiner is non-intrusive: it does not require any modification of existing software, or injection of network packets. More importantly, NSDMiner achieves higher accuracy than previous network-based approaches. Our experimental evaluation, which uses network traffic collected from our campus network, shows that NSDMiner outperforms the two best existing solutions by a significant margin.

Specifically, we developed three additional techniques to assist in the automatic identification of network service dependencies through passively monitoring and analyzing network traffic, including a logarithm-based ranking scheme aimed at more accurate detection of network service dependencies with lower false positives, an inference technique for identifying the dependencies involving infrequently used network services, and an approach for automated discovery of clusters of network services configured for load balancing or backup purposes. We performed extensive experimental evaluation of these techniques using real-world traffic collected from a college campus network. The experimental results demonstrate that these techniques advance the state of the art in automated detection and inference of network service dependencies.

5.6 Additional Research Accomplishments

The wide array of accomplishments described in the previous sections is not exhaustive of the work performed as part of the project mentioned earlier in this chapter. In the following, we briefly describe additional lines of research

we pursued and the accomplishments we achieved in those areas. We refer the reader to our previous publications for more information.

We studied network diversity as a security property of networks [16]. The interest in diversity as a security mechanism has recently been revived in various applications, such as Moving Target Defense (MTD), resisting worms in sensor networks, and improving the robustness of network routing. However, most existing efforts on formally modeling diversity have focused on a single system running diverse software replicas or variants. At a higher abstraction level, as a global property of the entire network, diversity and its impact on security have received limited attention. In our work, we took the first step towards formally modeling network diversity as a security metric for evaluating the robustness of networks against potential zero-day attacks. Specifically, we first devised a biodiversity-inspired metric based on the effective number of existing distinct resources. We then proposed two complementary diversity metrics, based on the least and average attacker's effort, respectively.

We also proposed a probabilistic framework for assessing the completeness and quality of available attack models [2], both at the intrusion detection level (e.g., IDS signatures) and at the alert correlation level (e.g., attack graphs). Intrusion detection and alert correlation are valuable and complementary techniques for identifying security threats in complex networks. However, both methods rely on models encoding a priori knowledge of either normal or malicious behavior. As a result, these methods are incapable of quantifying how well the underlying models explain what is observed on the network. Our approach overcomes this limitation, and enables us to estimate the probability that an arbitrary sequence of events is not explained by a given set of models. We leverage important mathematical properties of this framework to estimate such probabilities efficiently, and design fast algorithms for identifying sequences of events that are unexplained with a probability above a given threshold. This approach holds promise of identifying zero-day attacks, because such attacks (by definition of zero-day) are likely to be incompatible with all known traffic patterns.

Finally, we developed Switchwall [12], an Ethernet-based network fingerprinting technique that detects unauthorized changes to the L2/L3 network topology, the active devices, and the availability of an enterprise network. The network map is generated at an initial known state and is then periodically verified to detect deviations in a fully automated manner. Switchwall leverages a single vantage point and uses only very common protocols (PING and ARP) without any requirement for new software or hardware. Moreover, no previous knowledge of the topology is required, and our approach works on mixed speed, mixed vendors networks. Switchwall is able to identify a wide-range of changes, and this capability has been validated by our experimental results on both real and simulated networks.

6 Conclusions

As we discussed, the process of situation awareness in the context of cyber defense consists of three phases: situation perception, situation comprehension,

and situation projection. Situation awareness is generated and used across these three phases, and cyber analysts must answer several key questions during this process. In this chapter, we outlined an integrated approach to cyber situation awareness, and presented a framework – comprising several mechanisms and automated tools – that can help bridge the semantic gap between the available low-level data and the mental models and cognitive processes of security analysts.

In our project, we focused on techniques and tools for automatically answering the questions the analyst may ask about the current situation, the impact and evolution of an attack, the behavior of the attackers, the quality of available information and models, and the plausible futures of the current situation.

Although this framework represents a first important step in the right direction, a lot of work remains to be done for systems to achieve self-awareness capabilities. Key areas that need to be further investigated include adversarial modeling and reasoning under uncertainty, and promising approaches may include game-theoretic and control-theoretic solutions.

References

1. Albanese, M., Jajodia, S.: Formation of awareness. In: Kott, A., Wang, C., Erbacher, R.F. (eds.) *Cyber Defense and Situational Awareness. Advances in Information Security*, vol. 62, pp. 47–62. Springer, Cham (2014)
2. Albanese, M., Erbacher, R.F., Jajodia, S., Molinaro, C., Persia, F., Picariello, A., Sperli, G., Subrahmanian, V.: Recognizing unexplained behavior in network traffic. In: Pino, R.E. (ed.) *Network Science and Cybersecurity. Advances in Information Security*, vol. 55, pp. 39–62. Springer, New York (2014)
3. 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, June 2012
4. 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). doi:[10.1007/978-3-642-23822-2_23](https://doi.org/10.1007/978-3-642-23822-2_23)
5. 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 (2013)
6. 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, New York (2010)
7. 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)*, Baltimore, pp. 1339–1344, November 2011
8. 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, Massive Computing*, vol. 5, pp. 247–266. Springer, New York (2005)

9. Leversage, D.J., Byres, E.J.: Estimating a system's mean time-to-compromise. *IEEE Secur. Priv.* **6**(1), 52–60 (2008)
10. 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, October 2006
11. Natrajan, A., Ning, P., Liu, Y., Jajodia, S., Hutchinson, S.E.: NSDMine: automated discovery of network service dependencies. In: *Proceedings of the 31st Annual International Conference on Computer Communications (INFOCOM 2012)*, Orlando, pp. 2507–2515, March 2012
12. Nazzicari, N., Almillategui, J., Stavrou, A., Jajodia, S.: Switchwall: automated topology fingerprinting and behavior deviation identification. In: Jøsang, A., Samarati, P., Petrocchi, M. (eds.) *STM 2012. LNCS*, vol. 7783, pp. 161–176. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-38004-4_11](https://doi.org/10.1007/978-3-642-38004-4_11)
13. 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, October 2004
14. 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). doi:[10.1007/978-3-642-15497-3_35](https://doi.org/10.1007/978-3-642-15497-3_35)
15. Wang, L., Noel, S., Jajodia, S.: Minimum-cost network hardening using attack graphs. *Comput. Commun.* **29**(18), 3812–3824 (2006)
16. Wang, L., Zhang, M., Jajodia, S., Singhal, A., Albanese, M.: Modeling network diversity for evaluating the robustness of networks against zero-day attacks. In: Kutyłowski, M., Vaidya, J. (eds.) *ESORICS 2014. LNCS*, vol. 8713, pp. 494–511. Springer, Cham (2014). doi:[10.1007/978-3-319-11212-1_28](https://doi.org/10.1007/978-3-319-11212-1_28)

Theory and Models for Cyber Situation Awareness

Liu, P.; Jajodia, S.; Wang, C. (Eds.)

2017, VII, 227 p. 71 illus., Softcover

ISBN: 978-3-319-61151-8