Chapter 4
Uncertainty and Risk Management in Cyber Situational Awareness

Jason Li, Xinming Ou, and Raj Rajagopalan

Abstract Handling cyber threats unavoidably needs to deal with both uncertain and imprecise information. What we can observe as potential malicious activities can seldom give us 100% confidence on important questions we care about, *e.g.* what machines are compromised and what damage has been incurred. In security planning, we need information on how likely a vulnerability can lead to a successful compromise to better balance security and functionality, performance, and ease of use. These information are at best qualitative and are often vague and imprecise. In cyber situational awareness, we have to rely on such imperfect information to detect real attacks and to prevent an attack from happening through appropriate risk management. This chapter surveys existing technologies in handling uncertainty and risk management in cyber situational awareness.

4.1 Reasoning about Uncertainty is a Necessity

In the physical world it is commonplace that one must deal with uncertainty when security is concerned. For example, law enforcement agencies do not (and cannot) know every aspect of every individual citizen's life. But when a crime is committed there are effective investigative methods to capture the perpetrators. As a result we are not living in havoc where crimes are committed everywhere. Like the law enforcement agencies, cyber defenders also have to deal with a great deal of uncertainty, the degree of which is compounded by the nature of computing. For example, it is impossible for a system administrator to know what is going on within every computer inside an enterprise network. Even when every activity on every device can be logged, there is currently no effective way to process the logs due to their vagueness as attack indicators as well as the sheer volume of them. For example, a log showing SSH log in from an external IP address could be from a legitimate user, or from an adversary who has stolen a valid user credential. An HTTP packet overflowing a buffer in the web service could be due to an application error or an attempt to gain privilege on the server. Cyber defenders do not know who the attackers are nor where they are. Even with the help of intrusion detection systems (IDS), the large number of false positives brings significant uncertainty to the true interpretation of IDS alerts. And there are still false negatives where some attacks will not be reported by any IDS sensor. There are plenty of zero-day

Jason Li, Intelligent Automation, Inc. · Xinming Ou, Kansas State University · Raj Rajagopalan, HP Labs

vulnerabilities[1] in application software and there is no way to know for sure which software can be exploited by an attacker. With the large number of personal computers, laptops connected from home, and various emerging digital devices becoming part of enterprise networks, system administrators can no longer have a static picture of the network topology nor the precise configuration of every device in the network. The bottom line is, it is not possible for cyber defenders to be completely "certain" or "accurate" about all security-relevant information, yet they need to make decisions in the security management process, which is largely manual and ad-hoc these days. Unlike crimes in the physical world, automation has enabled cyber crimes to be conducted at a much higher speed and volume; without significant automation on the defense side we would not be able to effectively stem the threats.

The uncertainty challenge exists in all three phases of cyber situation awareness: prior security risk management, real-time intrusion detection, and posterior forensics analysis. The nature of uncertainty in these three aspects are slightly different. In risk management what we are uncertain about is the likelihood that a vulnerability exists in a piece of software, the chances that a vulnerability can be exploited successfully, the possibility that a user may succumb to social engineering, and so on. This type of uncertainty is in some sense "static" and reflects various kinds of risks inherent in a system. We call it the *static uncertainty*. The uncertainty in real-time situation awareness mostly arises from the invisibility of attackers in cyber space — it is hard to know where the attackers are and what choices he(she) has made in carrying out the attack. As a result all the IDS sensors can only capture the symptomatic phenomena caused by attacks but cannot positively ascertain whether an attack has happened and succeeded. The same problem exists for forensics analysis, with added difficulty caused by the gigantic amount of data but also more processing time available compared with the intrusion detection problem. We use the term "intrusion analysis" to encompass both the problem of intrusion detection and forensics analysis. We call the type of uncertainty found in intrusion analysis the *dynamic uncertainty*, since they are mostly related to dynamic events. We will focus our discussion on the dynamic uncertainty but will also briefly address the static uncertainty.

4.2 Two Approaches to Handling Dynamic Uncertainty

The challenge in handling dynamic uncertainty is how to start from imprecise and limited knowledge about attack possibilities, and quickly sift through large amounts of log information to identify a small set of data that altogether makes the picture of attacks clear. In doing so, the uncertainty about the system's security will be drastically reduced. For example, in many network intrusions, a small number of system logs are often sufficient to show that an attack has certainly happened, as well as how it progressed. The difficulty is how to start from uncertain views of the potential problem (*e.g., IDS alerts*) and quickly search for a few log entries from Terabytes of them so that the attacker's trace is clearly shown. System administrators are highly time-constrained. An automatic tool that can sift through the ocean of uncertainty to quickly and accurately locate the problem areas will be highly valuable in practice.

4.2.1 The logical approach

It has long been recognized that logical relations in computer attack conditions are important to consider in security analysis [5, 12, 52]. Modeling of such relations have yielded various ap-

[1] A zero-day vulnerability is one that has not been reported publicly but known by the underground hackers.

proaches to vulnerability analysis [3, 7, 10, 13, 18, 19, 25, 27, 36, 38, 39, 44, 45, 50, 51, 53] and IDS alert correlation [7, 9, 33, 35, 37, 54]. They have all adopted a somewhat deterministic logic in modeling: if the pre-condition of an attack is true, the post-condition is true. While these types of logical relations are important, they cannot account for the uncertainty in cyber security analysis. For example, an abnormal high network traffic is often an alert to system administrators on potential security problems. How can we model this in a deterministic logic? Does the observation reflect an attack activity? What is its pre- and post-conditions? It is hard to give definite answers to these questions because many events (both attack and non-attack activities) could cause a high network traffic. Another example is zero-day vulnerabilities, which have enabled a large number of intrusions into enterprise networks. One cannot make a deterministic judgment on whether a piece of software contains a zero-day vulnerability, but has to consider this possibility in security defense.

4.2.2 The statistical approach

A natural approach to handling uncertainty is to use statistical models, and there have been numerous attempts of this in the past [4, 11, 17]. However, there is a fundamental limitation in solving the uncertainty problems in cybersecurity using statistical models *alone*. Attackers do not play by rules. They adapt and do not typically follow a statistical pattern, as demonstrated by various forms of evading techniques [14, 15]. Thus, it is unlikely that statistical models alone can provide high-confidence conclusion on observed security events. Nevertheless, many such events have a statistical nature. A high network traffic deviating from the statistical norm gives a valuable hint on potential problems, and the confidence level on the true causes of such alerts can be statistically described as false positives and false negatives. It is important to account for the statistical differences in various assertions' confidence level. For example, compared with the anomalous high network traffic, a netflow filter that shows communication with known BotNet controllers is a more confident assertion on attacker activity. A simple and effective model for such statistical differences on assertion confidence will help in tackling the uncertainty problem.

To summarize, both deterministic logics and statistical models are valuable tools in cyber defense, but neither alone is sufficient to tackle the uncertainty challenge. Combining the two, however, will likely yield a reasoning method much more powerful than their sum. A reasoning framework that accounts for *both* logical relations *and* confidence differences among the various assertions will be the key in handling uncertainty in cybersecurity. How to design such a framework and apply it in security analysis is still an open problem. In the next two sections we will illustrate two recent attempts at achieving this "marriage" between logical causality and uncertainty. Section 4.3 describes an approach through statistical graphical model (Bayesian Network) derived from attack graphs [24]; Section 4.4 describes a variant of modal logic empirically developed from studying real intrusion incidents [40, 41].

4.3 From Attack Graphs to Bayesian Networks

To carry out enterprise security analysis, attack graphs have become the main-stream approach [3, 18–20, 26, 38, 50, 51, 53]. An attack graph illustrates all possible multi-stage attacks in an enterprise network, typically by presenting the logical causality relations among multiple privileges and configuration settings. Such logical relations are *deterministic*: the bad things will certainly happen in their worst forms as long as all the prerequisites are satisfied, and no bad things will happen if such conditions do not hold. While it is important to understand such logical relations, the deterministic nature has limited their use in practical network defense, especially when uncertainty has to been dealt with such as in intrusion detection and response.

4.3.1 A case study

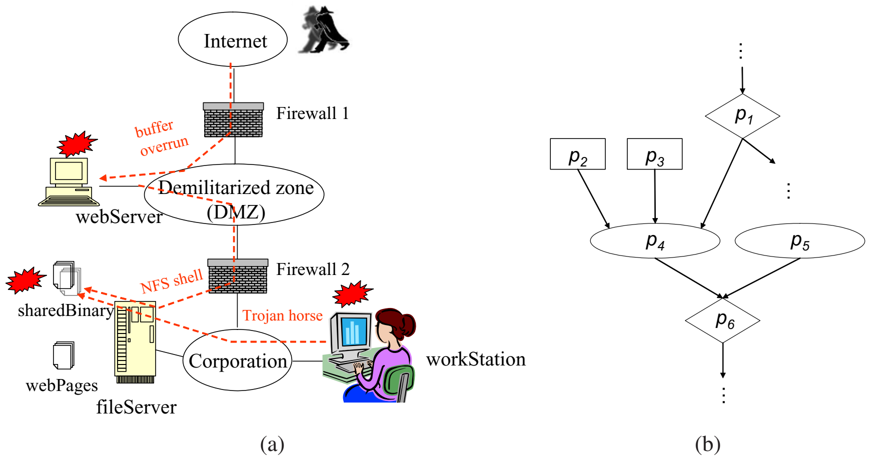


Fig. 4.1 An example attack scenario and attack graph.

Let us look at an example as shown in Figure 4.1, which is based on a real intrusion [38]. Suppose the following potential attack paths are discovered after analyzing the configuration. An attacker first compromises `webServer` by remotely exploiting vulnerability `CVE-2002-0392` to get local access on the server. Since `webServer` is allowed to access `fileServer` through the NFS protocol, he can then try to modify data on the file server. There are two ways to achieve this. If there are vulnerabilities in the NFS service daemons, he can try to exploit them and get local access on the machine; or if the NFS export table is not set up appropriately, he can modify files on the server through the NFS protocol by using programs like NFS Shell[2]. Once he can modify files on the file server, the attacker can install a Trojan-horse program in the executable binaries on `fileServer` that is mounted by machine `workStation`. The attacker can now wait for an innocent user on `workStation` to execute it and obtain control on the machine. Portion of the attack graph corresponding to the above scenario is shown in Figure 4.1 (b).

The node p_4 and its parents p_1, p_2, p_3 express the causality relation in the NFS Shell attack: if an attacker compromises the web server (p_1), the web server can access the file server through the NFS protocol (p_2), and the file server exports a partition to the web server (p_3), then the attacker will be able to launch an NFS Shell attack to access files on the file server (p_4). Suppose we want to use this piece of information in real-time security analysis. When we suspect the web server has been compromised, with how much confidence can we say that the files on the file server have been compromised? The answer is far less certain than the deterministic logic of the attack graph. How can we know whether the attacker has chosen to launch this attack? Even if he did so, how can we

[2] A program that provides user-level access to an NFS server (ftp://ftp.cs.vu.nl/pub/leendert/nfsshell.tar.gz)

know the attack has succeeded? Moreover, how can we account for the real-time observations that may be relevant to the question. For example, a file system integrity checker such as Tripwire [22] may report that certain files have been modified during the period. How shall we update our belief about possible attacks given this observation?

The problem of intrusion analysis is a far more imprecise process than deterministic reasoning. We do not know the attacker's choices thus there is the uncertainty from unknown attacker motivation and behaviors. Cyber attacks are not always 100% guaranteed to succeed thus there is the uncertainty from the imperfect nature of exploits. The defender's observations on potential attack activities are limited and as a result we have the uncertainty from false positives and false negatives of IDS sensors. Nevertheless, the logical causality encoded in a deterministic attack graph is invaluable to understand real-time security events, and will be useful for building practical network defense tools if we can *appropriately* account for the uncertainty inherent in the reasoning process.

4.3.2 Desired properties of Bayesian Networks in Intrusion Analysis

Recent years have seen a number of attempts at using Bayesian Networks to model such uncertainty in security analysis [2, 24, 32, 58]. Bayesian Network (BN) [43] marries the logical structure represented as a directed acyclic graph (DAG) and the uncertain nature of reasoning encoded as the conditional probability table (CPT) associated with every node in the graph. Using a BN model, one can query for questions like "*how likely a machine has been compromised given the current evidence*", "*based on the current observation, what is likely to happen next and what shall I do to prevent it from happening*", and "*which sensors shall be further investigated to confirm/rule out attacks*". This could yield powerful tools for real-time security analysis **if a BN model can be built that reflects reality**. Two key parts in building a BN are: 1) the graph structure and 2) the CPT parameters. Since attack graphs already provide a graphical model that reflects logical causality, it is natural to base the BN structure on the attack graph. How to obtain the CPT parameters has remained a difficult task. We believe the following are desirable properties of a BN model for cyber security analysis:

1. The graphical structure shall modularize and separate various types of uncertainty and avoid mingling different types of uncertainty in the same CPT.
2. The majority of CPT parameters shall be computed automatically from well-defined and realistic data sources.
3. The BN model shall not be too sensitive to perturbation on the CPT parameters.

Cyber security analysis, unlike other more well-behaved problem domains, does not naturally lend itself to statistical analysis. We do not have the ground truths in real traces from which we can learn the large number of CPT parameters, and the attackers are constantly adapting. As a result the CPT parameters need to be produced from often vague and subjective judgments. It is infeasible to ask a human user to assign every CPT parameter for every BN. The vast majority of these numbers need to be computed automatically from various data sources that reflect the various types of uncertainty in cyber security. A BN model that modularizes and separates the various types of uncertainty will make this process easier. Since those numbers are imprecise in nature, the result of BN analysis shall not be too sensitive to CPT parameters.

4.3.3 Building BN's from attack graphs

A Bayesian Network (BN) is a graphical representation of cause-and-effect relationships within a problem domain. More formally, a Bayesian network is a Directed Acyclic Graph (DAG) in which: the nodes represent variables of interest (propositions); the set of directed links represent the causal influence among the variables; the strength of an influence is represented by conditional probability tables (CPT). For example, if we imagine the graph structure in Figure 4.1 (b) is a Bayesian network, the node p_4 could have the following CPT associated with it.

p_1	p_2	p_3	p_4
T	T	T	0.8
otherwise			0

If all of p_4's parents are true, the probability of p_4 being true is 0.8. In all other cases the probability is 0 (p_4 is false). For any node in the DAG, given its parents, that node is conditionally independent of any other node that is not its descendent. This conditional independence makes a Bayesian network model a compact representation of the joint probability distribution over the interested variables. Bayesian networks can also serve as the inference engine, and can compute efficiently any queries over the variables modeled therein [43].

The semantics of BN's graph structure corresponds to that of an attack graph, especially a type of attack graphs called "logical attack graph" [38, 46] where each vertex is associated with a proposition and the arcs represent the logical causality relations between the vertices. We give the meaning of the propositions in the example of Figure 4.1.

p_1 : execCode(webServer,apache)
p_2 : reachable(webServer,fileServer, nfsProtocol,nfsPort)
p_3 : nfsExportInfo(fileServer,/export, write,webServer)
p_4 : nfsShellAttackAccomplished(fileServer, /export, write)
p_5 : localFileAccessAcomplished(fileServer, /export, write)
p_6 : accessFile(fileServer,write,/export)

The vertices are divided into three types. The square vertices are the ones that do not have any parent (*e.g.* p_2, p_3). They typically represent the input to the attack-graph generator — network reachability, configuration settings, potential vulnerabilities in software applications, security advisories, and so on. The diamond vertices represent privileges an attacker could obtain, such as user apache's privilege on the web server (p_1) and file modification privilege on the file server (p_6). Since there may be more than one way to obtain a privilege, the incoming arcs to a diamond vertex form a logical OR relation. For example, p_6 can be enabled by either p_4 or p_5. Thus, we call those vertices "OR node". The elliptic vertices, on the other hand, represent the logical AND relations and are called "AND node". Many attacks can only be accomplished when multiple preconditions are met, and the AND nodes capture this semantics. In the example, the attack p_4 can be accomplished only when all its three parents are true, and hence it is an AND node. Such OR and AND semantics can be easily encoded in the CPT's of a Bayesian network, which subsumes the deterministic logical relations. Thus there is no need to distinguish the AND/OR node types when we convert the attack graph to a BN.

If we want to construct a Bayesian Network from an attack graph for real-time security analysis, can we simply use the unmodified graph structure and attach CPT's to all the nodes to capture the uncertainty in reasoning? The answer is no. For example, we know that due to the uncertainty from attacker choice, p_4 may not become true after all of p_1, p_2, and p_3 are true simply because the attacker did not choose to launch the attack. To model this uncertainty under this unmodified graph structure, we would have to use the CPT associated with node p_4. However, there may be other reasons why p_4 does not become true after all its parents are true — for example, the attacker may have chosen to launch the attack but the attack failed due to the difficulty nature of the exploit. Such uncertainty arising from the inherent nature of a vulnerability will have to be encoded in the same CPT associated with p_4. Thus the CPT number 0.8 will have a number of contributing factors in it, which makes the generation and maintenance of the CPT parameters a difficult task.

For example, when we see the same attack activity in other parts of the network, we may want to increase the likelihood that an attacker may choose to use this attack. But in the unmodified graph structure there is no easy way to separate this attacker-choice uncertainty from the other factors in the CPT number of 0.8. As a result this type of correlation cannot be conducted elegantly. In this case the BN structure does not modularize various types of uncertainty into separate CPT's, and it violates principal 1 introduced in Section 4.3.2. This is just one example problem we have discovered in the current literature on building BN's from attack graphs for security analysis. We believe a more disciplined BN construction methodology needs to be studied to better capture the uncertainty in cyber security.

In summary, Bayesian Network provides a promising approach to handle uncertainty in cyber situational awareness. But key challenges still remain as to how to build/maintain the BN model efficiently. This is still a rapidly evolving field and interested readers are encouraged to consult the relevant literature cited in this chapter.

4.4 An Empirical Approach to Developing a Logic for Uncertainty in Situation Awareness

While statistical graphical models like Bayesian Network are theoretically rigorous and proven effective in other areas of research, when it comes to intrusion analysis they have an un-addressed gap, namely how to set statistical parameters in terms of hard probability distributions. For security analysis, it is nearly impossible to obtain the ground truth in real traces and it is hard if not impossible to realistically simulate attack scenarios. Similarly, while it would be ideal to characterize intrusion analysis tools in terms of hard metrics such as alarm compression ratio combined with true and false positive ratios (see [54] for definitions and other metrics), it is impossible to calibrate tools without representative data with known ground truth. At the same time, the fact that human system administrators have been using manual analysis and low-level tools to detect attacks in logs and real-time alerts inspires us to formulate a logic, *in an empirical manner*, that approximates human reasoning that works with a qualitative assessment on a few confidence levels that are relatively easy to understand. We acknowledge that this formulation not only hides the lack of knowledge of base probabilities but also reflects the great deal of ambiguity that exists in intrusion analysis of real data. We hope that by creating an option to specify the confidence level explicitly and by providing general practical tools to manipulate these uncertain pieces of knowledge, we can bypass some of these fundamental problems and gain experience that may make some statistical approaches viable in the future.

4.4.1 A case study

We first show a case study we conducted by interviewing a system administrator for a university campus network. He told us about how he identified compromised machines on the campus network during a security incident, which was due to a zero-day vulnerability in the TrendMicro anti-malware service. All the reasoning and correlation was done manually but was very effective. We then identify the rationale behind the decisions he made at various points, and design a logic that captures this reasoning process. The logic is capable of handling uncertainty which is crucial for real-time security analysis, since the observations are often vague and limited, and we do not know where the attacker is and what choices he made in an attack.

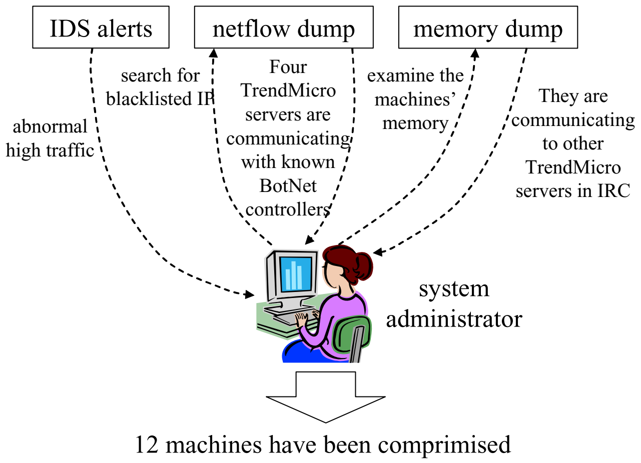
4.4.1.1 Scenario description

The scenario is illustrated in Figure 4.2(1), and described below. System Administrator (SA) noticed an abnormal increase in campus-network traffic (*Observation 1*). SA took the netflow dump for that time period and ran a packet capture tool on it to search for known malicious IP addresses, and identified that four Trend Micro servers initiated IRC connection to some known BotNet controllers (*Observation 2*). SA hence determined that the four TrendMicro servers likely had been compromised. He sat down at one of them and dumped the memory, from which he found what appeared to be malicious code (*Observation 3*). He also looked up for all the open TCP socket connections and noticed that the server had been connecting to some other Trend Micro servers on campus through the IRC channel (*Observation 4*). He hence determined that those servers were also compromised. He did the same thing for the other identified servers and found more compromised servers. Altogether he identified 12 compromised machines and took them off line.

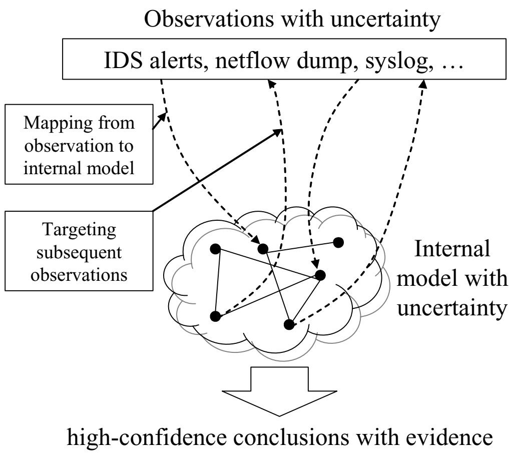
We observe that all the alerts above contain some amount of uncertainty regarding their implication. Observation 1 could just mean some users started downloading movies through BitTorrent. Observation 2 has a higher degree of likelihood that the identified servers are compromised, but simply an IRC connection to a known BotNet controller does not necessarily mean the machine has been compromised. For example, it could be that some system admin was probing BotNet controllers for research purposes. (The interviewed system administrator actually does this himself.) Observation 3 is also a strong indication that the machine has been controlled by an attacker. But it is not always easy to determine whether a suspicious module found in the memory dump is indeed malicious, especially with zero-day vulnerabilities as in this case. So this alert also contains some amount of false positive. Observation 4, like observation 2, cannot directly prove that the machines observed are under the control of attackers. However, when we put all the four pieces of evidence together, it becomes clear that an attack has certainly happened and succeeded, and we can tell with almost certainty which machines have been compromised. We observe two key components in human reasoning on uncertain information: 1) from the current beliefs based on the current observation, use logical reasoning to determine what additional observation could help to "strengthen" the beliefs' confidence; 2) from a large number of possibilities, derive a high-confidence belief corroborated by a number of complementary evidences logically linked together. For example, even though the SA was not sure whether the abnormal high traffic really indicated an attack, he knew that this observation is logically linked to network activity, which can also be observed by netflow dump. When from the netflow dump the TrendMicro servers were shown to communicate with malicious IP addresses, and from the memory dump a potentially malicious code module was found, the two pieces of evidence both indicated that the server was likely compromised, thus strengthening the belief's confidence to almost certain.

4.4.2 Encoding the case study in logic

Figure 4.2(2) presents a high-level view of the reasoning framework. The framework consists of two layers: observations and an internal reasoning model, both with uncertainty. The observations are from system monitoring data such as IDS alerts, netflow dumps, syslog, *etc.* They are mapped into the internal reasoning model as conditions representing unobservable security status under interest. For example, `abnormal high traffic` is an observation, and `an attacker is performing some network activity` is an internal condition. Logical relations exist between observations and internal conditions with varying degrees of certainty. For example, we can say `abnormal high traffic indicates the possibility that an attacker is performing some network activity`. Another example: `netflow dump showing a host communicating with known BotNet controllers indicates that an attacker likely has compromised the host.` Likewise, there are logical relations among the various internal conditions and these relations contain


IDS alerts
netflow dump
memory dump
search for blacklisted IP
abnormal high traffic
Four TrendMicro servers are communicating with known BotNet controllers
examine the machines' memory
They are communicating to other TrendMicro servers in IRC
system administrator
12 machines have been comprimised

(1) Case scenario


Observations with uncertainty
IDS alerts, netflow dump, syslog, …
Mapping from observation to internal model
Targeting subsequent observations
Internal model with uncertainty
high-confidence conclusions with evidence

(2) Vision of applying a logic with uncertainty to real-time security analysis

Fig. 4.2 Case study and vision for automated situation awareness

varying degrees of certainty as well. For example, one can say `after an attacker compromised a machine, he may possibly perform some network activity from the machine`, and `after an attacker sends an exploit to a machine, he will likely compromise the machine`. These statements capture the rationale behind human reasoning when manually detecting intrusions from the logs. The internal model is analogous to a human brain — observations are reflected into a human's thinking as beliefs with varying strengths, and the beliefs are related to one another with varying strengths as well. We design logical rules to capture both aspects and a formalized reasoning process to simulate human's thinking such that 1) we can target observations to a small subset of all possible data through hints provided by the internal reasoning process; 2) we can sift through the uncertain observations and internal conditions to reach high-confidence beliefs for situation awareness.

4.4.2.1 Notations

There are two types of facts in our logic: those for observations and those for internal conditions. We use $obs(F)$ to denote a fact about observations, and $int(F)$ to denote a fact about internal conditions. For example,
$obs(netflowBlackListFilter(172.16.9.20, 129.7.10.5))$ is an observation from the netflow blacklist filter that machine 172.16.9.20 is communicating with a known malicious IP address 129.7.10.5, and $int(compromised(172.16.9.20))$ is an internal condition that machine 172.16.9.20 has been compromised. We use three modality operators p, l, c, standing for "possible, likely, certain" to express different confidence levels. p represents low confidence, l represents high confidence, and c is the almost certainty which is the goal of situation awareness. The logic formulation consists of two parts: *observation correspondence* which maps observations to internal conditions, and *internal model* that captures logical relations among internal conditions. The two parts for this example are shown in Figure 4.3 and 4.4 respectively.

$$\begin{aligned}
&A_1 : obs(anomalyHighTraffic) \xmapsto{p} int(attackerNetActivity)\\
&A_2 : obs(netflowBlackListFilter(H, BlackListedIP)) \xmapsto{l} int(attackerNetActivity)\\
&A_3 : obs(netflowBlackListFilter(H, BlackListedIP)) \xmapsto{l} int(compromised(H))\\
&A_4 : obs(memoryDumpMaliciousCode(H)) \xmapsto{l} int(compromised(H))\\
&A_5 : obs(memoryDumpIRCSocket(H_1, H_2)) \xmapsto{p} int(exchangeCtlMessage(H_1, H_2))
\end{aligned}$$

Fig. 4.3 Observation correspondence

4.4.2.2 Observation correspondence

Observation correspondence gives a "meaning" to an observation, in the form of internal conditions. For example, in A_1 an abnormal high network traffic (*anomalyHighTraffic*) is mapped to *int*(*attackerNetActivity*), meaning an attacker is performing some network activity. This is a low-confidence judgment thus the modality operator is p. Intuitively the p mode means there is other equally possible interpretations for the same observation. Rule A_2 and A_3 give the meaning to an alert identified in a netflow analysis. There are a number of filtering tools available that can search for potential malicious patterns in a netflow dump, such as "capture daemon" and "flow-nfilter". This rule deals with one filter that identifies communication with known malicious IP addresses. Since any such activity is a strong indication of attacker activity and the fact that the machine involved has been compromised, the modality of the two rules is l. There are still other possibilities, *e.g.*, the communication could be issued by a legitimate user who wants to find out something about the malicious IP address. But the likelihood of that is significantly lower than what is represented by the righthand side of the two rules. Rule A_4 says if memory dump on machine H identifies malicious code, then H is likely to be compromised. Rule A_5 says that if the memory dump identifies open IRC sockets between machine H_1 and H_2, then it is possible that the IRC channel is used to exchange control messages between BotNet machines.

We present these observation correspondence rules not to expect everyone would agree on them. For example, some reader may think the modality of rule A_4 shall be c. This is completely OK. One advantage of such a logic is that it facilitates discussion and sharing of security knowledge. Empirical experiences from a large community can help tune the modality parameters in those rules. We envision a rule repository model like that for Snort[3], where a community of participants contribute and agree upon a set of rules in an open language. Currently there are only coarse-grained classification and some natural-language explanations for the meanings behind each Snort alert. The logic language can be used by the Snort rule writers to develop a number of internal-model predicates so that the observation correspondence rules provide meanings to the various IDS alerts, which can then be reasoned about automatically.

$$\begin{aligned}
&I_1 : \mathit{int}(\mathit{compromised}(H_1)) \xrightarrow{pc} \mathit{int}(\mathit{sendExploit}(H_1,H_2))\\
&I_2 : \mathit{int}(\mathit{sendExploit}(H_1,H_2)) \xrightarrow{lp} \mathit{int}(\mathit{compromised}(H_2))\\
&I_3 : \mathit{int}(\mathit{compromised}(H_1)), \mathit{int}(\mathit{compromised}(H_2)) \xrightarrow{pc} \mathit{int}(\mathit{exchangeCtlMessage}(H_1,H_2))
\end{aligned}$$

Fig. 4.4 Internal model

4.4.2.3 Internal model

The $\xrightarrow{m_1 m_2}$ operator in the internal model represents the "leads to" relationship between internal conditions, and as a result the arrow must be aligned with time: since the righthand side is caused by the lefthand side, it must happen no earlier than the lefthand side. However, the *reasoning* can go along both directions, and hence two modality operators (m_1, m_2) are associated with each rule. For example, rule I_1 says "if an attacker has compromised machine H_1, he can send an exploit from H_1 to another machine H_2." The forward reasoning has a low certainty: the attacker may or may not choose to send an exploit after compromising a machine. Thus the forward reasoning is qualified by the p modality. Reasoning on the reverse direction however has the c modality: if an attacker sends an exploit from one machine to another, he shall have already compromised the first machine. In rule I_2, the fact that an attacker sends an exploit to a machine leads to the compromise of the machine. The forward reasoning is also not certain, since the exploit may fail to execute on the target host. We have used the confidence level l — attackers typically make sure their exploit will likely work before using it in an attack. On the other direction, sending in a remote exploit is just one of many possible ways to compromise a host. Thus the reverse reasoning for I_2 has the p modality. I_3 is the only rule in this model that has two facts on the lefthand side. Like in typical logic-programming languages, the comma represents the AND logical relation. The forward direction has the p modality: if an attacker compromised two machines H_1, H_2, the two machines can possibly exchange BotNet control messages between them to coordinate further attacks. The backward direction has the c modality: if two machines are exchanging BotNet control messages, both of them must have been compromised.

The above logical formulation is encoded in Prolog and a preliminary deduction system is implemented to simulate the "confidence strengthening" process in human reasoning. The key deduction rule for this is shown below.

$$\frac{\mathit{int}(F,l) \Leftarrow \mathit{Pf}_1 \quad \mathit{int}(F,l) \Leftarrow \mathit{Pf}_2 \quad \mathit{Pf}_1 \parallel \mathit{Pf}_2}{\mathit{int}(F,c) \Leftarrow \mathit{strengthenedPf}(\mathit{Pf}_1,\mathit{Pf}_2)}$$

We use $\mathit{int}(F,m) \Leftarrow \mathit{Pf}$ to indicate the internal fact F is true with modality m, and Pf is the proof which shows the derivation steps in the logic arriving at the conclusion. The $\parallel$ relation indicates two proofs are *independent*, which is defined as sharing no common observations and internal

[3] A widely used network intrusion detection system: `http://www.snort.org/`

facts. This deduction rule states that if we have two disjoint reasoning traces to reach a fact with confidence level *l*, then the fact's confidence level can be boosted to *c*. The application of this simple reasoning engine has already shown interesting results [41].

This is just the first small step towards achieving the vision presented in Figure 4.2. There are still great challenges ahead and significant research efforts are needed to tackle these challenges. Detailed technical description of the approach can be found in some recent manuscripts [40, 41].

4.4.3 Comparison with previous approaches

There have been numerous approaches to intrusion alerts correlation proposed in the past [7, 9, 33, 35, 37, 54]. They provide important insights into logical causality relations in IDS alerts, but fall short of a systematic solution to the real-time security analysis problem due to: 1) the lack of capabilities for representing varying degrees of uncertainty in assertions derived from system monitoring data and using these uncertain judgments to reach (almost) certain conclusions, and 2) the lack of a unified framework capable of handling *any* security-relevant information — not just IDS alerts.

The confidence level about "true" system conditions derived from an observation varies depending on the nature of the observation. A key challenge in situation awareness is how to sift through the ocean of uncertain observation and reach a conclusion with high confidence. To this end it is necessary to distinguish the different confidence levels in the reasoning process. Zhai *et al.* [58] uses Bayesian networks to correlate complementary intrusion evidence from both IDS alerts and system monitoring logs so that highly confident traces can be distinguished from less confident ones. However, one big question in such statistical models is how to obtain the numerical parameters, such as the conditional probability tables (CPT) associated with every node in a BN which may affect the analysis result. The paper presented a formulation for computing the CPT entries but it is far from clear how the computation can be justified theoretically or validated empirically. In comparison, the approach discussed above represents the various certainty levels qualitatively as modality operators, and the uncertainty levels are directly integrated into the reasoning process. As a result the reasoning system will have more leverage on utilizing the varying degrees of uncertainty to make better decisions.

Another drawback of the previous approaches is that the reasoning models often center around IDS alerts with pre- and postconditions. This will limit the model's capability to reason about a larger variety of system monitoring data. As illustrated in our case study, in some intrusions IDS alerts only play the role of triggering investigation. Sometimes there is no IDS alert at all either because there are no IDS devices deployed or the IDS devices missed the attack completely. Yet the system administrator was still able to trace to the compromised machines. Moreover, in our case study we found it very difficult to come up with a pre- and postcondition model for every possible observation. For example, what shall be the pre- and postconditions for an abnormal high network traffic? There are too many possibilities and a single pre- and postcondition won't be sufficient. Centering the reasoning model around IDS alerts with pre- and postconditions will likely become inflexible when more types of system monitoring information need to be included in the reasoning process. In comparison, the observation correspondence model in our logic gives a meaning to *any* observation (not just IDS alert). This model is more direct: it gives an observation a meaning about itself, instead of a meaning in terms of other conditions that can be logically linked to it. Another general model for incorporating all possible types of data in security analysis is M2D2 [33]. Compared with M2D2, our model is much more simple. For example, we do not classify information into various categories with mathematical notations such as functions and relations. Rather, everything in our model is just a simple "statement" — one that can be easily translated into natural language. Such simplicity in modeling has shown the advantage in incorporating a large variety of security-relevant information. Also M2D2 is not able to represent the varying degrees of uncertainty levels useful for security analysis. Given the complexity of cyber attacks and the large number of system monitoring tools, it is important to have a simple logic that can encode all possible types of security relevant information and capture the uncertain nature of such data's meanings in security analysis.

The limitation of statistical models has been discussed before. Instead of directly applying these existing models, we propose to start from the bottom and design a logic that approximates human reasoning. Unlike quantitative probabilities, the qualitative assessment on confidence levels such as "possible", "likely", and "certain" are relatively easy to be understood and used by human experts. As the first step we design a logic with *modality operators*[4] that capture the various qualitative confidence levels in reasoning about situation awareness. It is possible that this logic derived from empirical study can be refined and re-formalized in some existing statistical models described above, the well-studied theories on modal logic [23], or a combination of both. Another advantage of the logical approach is that the justification for a conclusion can be generated in the form of logical proofs. This makes it clear what conditions are used and what assumptions are made in reaching the conclusion. How to produce such justification is far less obvious in statistical models.

4.5 Static Uncertainty and Risk Management

Another important aspect of uncertainty in cyber situation awareness is the static uncertainty or the inherent risk in a system. Analysis and management of such risks is especially crucial to make sensible decisions in security investment. This has become a critical challenge in security management since enterprise networks have become essential to the operation of companies, laboratories, universities, and government agencies. As they continue to grow both in size and complexity, vulnerabilities are regularly discovered in software applications which are exploited to stage cyber attacks. Currently, management of security risk of an enterprise network is an art and not a science. There is no objective way to measure the security of an enterprise network. As a result it is difficult to answer such questions as "are we more secure than yesterday", or "how should we invest our limited resources to improve security", or "how does this vulnerability impact the overall security of my system". By increasing security spending an organization can decrease the risk associated with security breaches. However, to do this tradeoff analysis there is a need for quantitative models of security instead of the current qualitative models.

The issue of security metrics has long attracted much attention [21, 29, 30], and recent years have seen significant effort on the development of quantitative security metrics [1, 6, 8, 28, 34, 42, 47]. However, we are yet to establish metrics that can objectively quantify security risk. Part of the reason is that the treatment on many aspects of security still stays at the deterministic level. For example, resistance to attacks is often regarded as binary: an attack is either impossible or trivial. This is remote from realistic security management, where one has to admit the varying degrees of exploit difficulty levels to distinguish more imminent threats from less ones.

4.5.1 CVSS metrics

The Common Vulnerability Scoring System (CVSS) [31, 49] provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. CVSS consists of three metric groups: Base, Temporal and Environmental. Each of this group produces a vector of compressed textual representation that reflects various properties of the vulnerability (*metric vector*). A formula takes the vector and produces a numeric score in the range of 0 to 10 indicating the severity of the vulnerability. The most important contribution of CVSS is the metric vector: it provides a wide

[4] Similar to the modality operators in modal logic [23].

range of vulnerability properties that are the basis for deriving the numerical scores. We use the term "CVSS metrics" to denote the metric vectors and the term "CVSS scores" to denote the numerical scores. Any CVSS score must be published along with the CVSS metrics. The CVSS metrics convey much richer information than the numerical scores and are closer to measurable properties of a vulnerability.

AC metric	*Description*
High	Specialized access conditions exist.
Medium	The access conditions are somewhat specialized.
Low	Specialized access conditions or extenuating circumstances do not exist.

(a) The CVSS Access Complexity Metric

E metric	*Description*
Unproven (U)	No exploit code is available, or an exploit is entirely theoretical.
Proof-of- Concept (POC)	Proof-of-concept exploit code or an attack demonstration that is not practical for most systems is available. The code or technique is not functional in all situations and may require substantial modification by a skilled attacker.
Functional (F)	Functional exploit code is available. The code works in most situations where the vulnerability exists.
High (H)	Either the vulnerability is exploitable by functional mobile autonomous code, or no exploit is required (manual trigger) and details are widely available. The code works in every situation, or is actively being delivered via a mobile autonomous agent (such as a worm or virus).

(b) The CVSS Exploitability Metric

Table 4.1 Example CVSS metrics

The Base group represents the intrinsic and fundamental characteristics of the vulnerability that are constant over time and user environments. The Temporal group represents the characteristics of vulnerability that change over time but not among user environments. The Environmental group represents the characteristics of a vulnerability that are relevant and unique to a particular user environment. Generally, the base and temporal metrics are specified by vulnerability bulletin analysts, security product vendors, or application vendors because they typically have better information about the characteristics of a vulnerability and the most up-to-date exploit technology than do users. The environmental metrics, however, are specified by users because they are best able to assess the potential impact of a vulnerability within their own environments. We observe that how a vulnerability may impact the security of an enterprise network can only be assessed when the security interactions among components of the network are taken into account. Since attack graphs capture security interactions within an enterprise network, it is possible that the Basic and Temporal metrics can be used in combination with attack graphs to compute a cumulative risk metric for an enterprise network.

As some examples, the *Access Complexity (AC)* metric in the Base group and the *Exploitability (E)* metric in the Temporal group are shown in Table 4.1. The AC metric gauges the complexity of exploiting a vulnerability based on whether special conditions, *e.g.* certain race conditions, are necessary to successfully exploit the vulnerability. A "High" AC metric indicates a vulnerability is inherently difficulty to exploit, and thus will likely have a low success likelihood. The E metric measures the current state of exploit techniques or code availability. Public availability of easy-to-use exploit code increases the number of potential attackers by including those who are unskilled, thereby increasing the vulnerability's overall success likelihood.

4.5.2 Combining CVSS and Attack Graphs

The main limitation of CVSS is that it provides scoring of individual vulnerabilities but it does not provide a sound methodology for aggregating the metrics for a set of vulnerabilities in a network to provide an overall network security score. For example, there can be a situation in which the individual vulnerability scores are low but they can be combined to compromise a critical resource. The overall security of a network configuration running multiple services cannot be determined by simply counting the number of vulnerabilities or adding up the CVSS scores. Attack graphs can be used to model causal relationship among vulnerabilities. Recent years have seen a number of attempts at calculating quantitative metrics by combining CVSS metrics on attack graphs [16, 48, 55–57]. We expect that these models can also be used to suggest configuration changes that mitigate the threats to an enterprise.

4.6 Conclusion

Both aspects of uncertainty: static uncertainty and dynamic uncertainty, are important in cyber situational awareness. We discussed two promising approaches to addressing the dynamic uncertainty challenge, and briefly introduced the CVSS metrics which can be used in combination with attack graphs to address the static uncertainty challenge for automated risk management. The research for systematically handling uncertainty and risk in cyber space is still in a preliminary stage. However, without addressing this problem in a scientific manner, it will be difficult to achieve sustainable cyber defense.

References

1. Ehab Al-Shaer, Latif Khan, and M. Salim Ahmed. A comprehensive objective network security metric framework for proactive security configuration. In *ACM Cyber Security and Information Intelligence Research Workshop*, 2008.
2. Magnus Almgren, Ulf Lindqvist, and Erland Jonsson. A multi-sensor model to improve automated attack detection. In *11th International Symposium on Recent Advances in Intrusion Detection (RAID 2008)*. RAID, September 2008.
3. Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of 9th ACM Conference on Computer and Communications Security*, Washington, DC, November 2002.
4. Stefan Axelsson. A preliminary attempt to apply detection and estimation theory to intrusion detection. Technical report, Chalmers Univ. of Technology, 2000.
5. R. Baldwin. Rule based analysis of computer security. Technical Report TR-401, MIT LCS Lab, 1988.
6. Davide Balzarotti, Mattia Monga, and Sabrina Sicari. Assessing the risk of using vulnerable components. In *Proceedings of the 2nd ACM workshop on Quality of protection*, 2005.
7. Steven Cheung, Ulf Lindqvist, and Martin W Fong. Modeling multistep cyber attacks for scenario recognition. In *DARPA Information Survivability Conference and Exposition (DISCEX III)*, pages 284–292, Washington, D.C., 2003.
8. Elizabeth Chew, Marianne Swanson, Kevin Stine, Nadya Bartol, Anthony Brown, and Will Robinson. *Performance Measurement Guide for Information Security*. National Institute of Standards and Technology, July 2008. NIST Special Publication 800-55 Revision 1.
9. Frédéric Cuppens and Alexandre Miège. Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy*, 2002.

10. J. Dawkins and J. Hale. A systematic approach to multi-stage network attack analysis. In *Proceedings of Second IEEE International Information Assurance Workshop*, pages 48 – 56, April 2004.
11. Dorothy Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2), 1987.
12. Daniel Farmer and Eugene H. Spafford. The COPS security checker system. Technical Report CSD-TR-993, Purdue University, September 1991.
13. William L. Fithen, Shawn V. Hernan, Paul F. O'Rourke, and David A. Shinberg. Formal modeling of vulnerabilities. *Bell Labs technical journal*, 8(4):173–186, 2004.
14. Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: Formal reasoning and practical techniques. In *Proceedings of The 13th ACM Conference on Computer and Communications Security (CCS 2006)*, Alexandria, VA, October 2006.
15. Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. Polymorphic blending attacks. In *Proceedings of The 15th USENIX Security Symposium*, Vancouver, B.C., Canada, August 2006.
16. Marcel Frigault, Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using dynamic bayesian network. In *Proceedings of the 4th ACM workshop on Quality of protection*, 2008.
17. Paul Helman and Gunar Liepins. Statistical foundations of audit trail analysis for the detection of computer misuse. *IEEE Transactions on Software Engineering*, 19(9), 1993.
18. Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. In *22nd Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, Florida, December 2006.
19. Sushil Jajodia, Steven Noel, and Brian O'Berry. Topological analysis of network attack vulnerability. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats: Issues, Approaches and Challanges*, chapter 5. Kluwer Academic Publisher, 2003.
20. Somesh Jha, Oleg Sheyner, and Jeannette M. Wing. Two formal analyses of attack graphs. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, pages 49–63, Nova Scotia, Canada, June 2002.
21. Daniel Geer Jr., Kevin Soo Hoo, and Andrew Jaquith. Information security: Why the future belongs to the quants. *IEEE SECURITY & PRIVACY*, 2003.
22. Gene H. Kim and Eugene H. Spafford. The design and implementation of tripwire: A file system integrity checker. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security (CCS)*, 1994.
23. Kenneth Konyndyk. *Introductory Modal Logic*. University of Notre Dame Press, 1986.
24. Jason Li, Peng Liu, and Xinming Ou. Using Bayesian Networks for cyber security analysis. Manusrcipt, 2008.
25. Wei Li, Rayford B. Vaughn, and Yoginder S. Dandass. An approach to model network exploitations using exploitation graphs. *SIMULATION*, 82(8):523–541, 2006.
26. Richard Lippmann, Kyle Ingols, Chris Scott, Keith Piwowarski, Kendra Kratkiewicz, Mike Artz, and Robert Cunningham. Validating and restoring defense in depth using attack graphs. In *Military Communications Conference (MILCOM)*, Washington, DC, U.S.A., October 2006.
27. Richard Lippmann and Kyle W. Ingols. An annotated review of past papers on attack graphs. Technical report, MIT Lincoln Laboratory, March 2005.
28. Pratyusa Manadhata, Jeannette Wing, Mark Flynn, and Miles McQueen. Measuring the attack surfaces of two FTP daemons. In *Proceedings of the 2nd ACM workshop on Quality of protection*, 2006.
29. John McHugh. Quality of protection: measuring the unmeasurable? In *Proceedings of the 2nd ACM workshop on Quality of protection (QoP)*, Alexandria, Virginia, USA, 2006.
30. John McHugh and James Tippett, editors. *Workshop on Information-Security-System Rating and Ranking (WISSRR)*. Applied Computer Security Associates, May 2001.
31. Peter Mell, Karen Scarfone, and Sasha Romanosky. *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*. Forum of Incident Response and Security Teams (FIRST), June 2007.

32. Gaspar Modelo-Howard, Saurabh Bagchi, and Guy Lebanon. Determining placement of intrusion detectors for a distributed application through bayesian network modeling. In *11th International Symposium on Recent Advances in Intrusion Detection (RAID 2008)*. RAID, September 2008.
33. Benjamin Morin, Hervé, and Mireille Ducassé. M2d2: A formal data model for ids alert correlation. In *5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, pages 115–137, 2002.
34. National Institute of Standards and Technology. *Technology assessment: Methods for measuring the level of computer security*, 1985. NIST Special Publication 500-133.
35. Peng Ning, Yun Cui, Douglas Reeves, and Dingbang Xu. Tools and techniques for analyzing intrusion alerts. *ACM Transactions on Information and System Security*, 7(2):273–318, May 2004.
36. Steven Noel, Sushil Jajodia, Brian O'Berry, and Michael Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *19th Annual Computer Security Applications Conference (ACSAC)*, December 2003.
37. Steven Noel, Eric Robertson, and Sushil Jajodia. Correlating intrusion events and building attack scenarios through attack graph distances. In *20th Annual Computer Security Applications Conference (ACSAC 2004)*, pages 350– 359, 2004.
38. Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *13th ACM Conference on Computer and Communications Security (CCS)*, pages 336–345, 2006.
39. Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. MulVAL: A logic-based network security analyzer. In *14th USENIX Security Symposium*, 2005.
40. Xinming Ou, Raj Rajagopalan, and Sakthiyuvaraja Sakthivelmurugan. A practical approach to modeling uncertainty in intrusion analysis. Technical report, Department of Computing and Information Sciences, Kansas State University, 2008.
41. Xinming Ou, S. Raj Rajagopalan, Abhishek Rakshit, and Sakthiyuvaraja Sakthivelmurugan. An empirical approach to modeling uncertainty in intrusion analysis. Under review, February 2009.
42. Joseph Pamula, Sushil Jajodia, Paul Ammann, and Vipin Swarup. A weakest-adversary security metric for network configuration security analysis. In *Proceedings of the 2nd ACM workshop on Quality of protection*, 2006.
43. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 1999.
44. Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *NSPW '98: Proceedings of the 1998 workshop on New security paradigms*, pages 71–79. ACM Press, 1998.
45. C. R. Ramakrishnan and R. Sekar. Model-based analysis of configuration vulnerabilities. *Journal of Computer Security*, 10(1-2):189–209, 2002.
46. Diptikalyan Saha. Extending logical attack graphs for efficient vulnerability analysis. In *Proceedings of the 15th ACM conference on Computer and Communications Security (CCS)*, 2008.
47. Mohamed Salim, Ehab Al-Shaer, and Latif Khan. A novel quantitative approach for measuring network security. In *INFOCOM 2008 Mini Conference*, 2008.
48. Reginald Sawilla and Xinming Ou. Identifying critical attack assets in dependency attack graphs. In *13th European Symposium on Research in Computer Security (ESORICS)*, Malaga, Spain, October 2008.
49. Mike Schiffman, Gerhard Eschelbeck, David Ahmad, Andrew Wright, and Sasha Romanosky. *CVSS: A Common Vulnerability Scoring System*. National Infrastructure Advisory Council (NIAC), 2004.
50. Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 254–265, 2002.

51. Laura P. Swiler, Cynthia Phillips, David Ellis, and Stefan Chakerian. Computer-attack graph generation tool. In *DARPA Information Survivability Conference and Exposition (DISCEX II'01)*, volume 2, June 2001.
52. Steven J. Templeton and Karl Levitt. A requires/provides model for computer attacks. In *Proceedings of the 2000 workshop on New security paradigms*, pages 31–38. ACM Press, 2000.
53. T. Tidwell, R. Larson, K. Fitch, and J. Hale. Modeling Internet attacks. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, West Point, NY, June 2001.
54. Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, 2004.
55. Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An attack graph-based probabilistic security metric. In *Proceedings of The 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC'08)*, 2008.
56. Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using attack graphs. In *Third Workshop on Quality of Protection (QoP)*, 2007.
57. Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring the overall security of network configurations using attack graphs. In *Proceedings of 21th IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC'07)*, 2007.
58. Yan Zhai, Peng Ning, Purush Iyer, and Douglas S. Reeves. Reasoning about complementary intrusion evidence. In *Proceedings of 20th Annual Computer Security Applications Conference (ACSAC)*, pages 39–48, December 2004.