Investigative Response Modeling and Predictive Data Collection

Dan Moor
HP
dan.moor@hp.com

S. Raj Rajagopalan
Honeywell ACS
siva.rajagopalan@honeywell.com

Sathya Chandran Sundaramurthy, Xinming Ou
Kansas State University
{sathya, xou}@ksu.edu

***Abstract*—While most enterprise computing environments are proactively monitored for threats and security violations using automated detection engines, the ability to validate reported events as true incidents still requires a non-trivial amount of time and information gathering as well as investment in staffing and training of personnel. To improve an organization's overall reactive security posture and reduce some of the associated costs we propose an investigation model supported by predictive, automated data collection and guided presentation of the resulting information. By modeling the investigative goals and requirements for each event type, this approach can automate proactive data collection actions wherever possible thus reducing the investigation time as well as providing a consistent framework for the monitoring staff. By providing the goals of the alert validation process the framework also reduces the minimum skill required of monitoring staff. Furthermore, the collected information is presented in a formatted manner with documented requirements for validation therefore guiding the analyst to the appropriate conclusion. By following this method, false positive alerts are more quickly pared down allowing for better utilization of skilled resources by focusing efforts on only those alerts validated as genuine.**

***Index Terms*-Big data, computer forensics, digital investigation, incident response, predictive modeling.**

I. Introduction

Host and network based security monitoring tools have evolved into robust and comprehensive solutions providing high levels of visibility in understanding the state of security for their respective monitored resources. These solutions are prone to high rates of false alarms requiring a non-trivial investment in personnel and effort by the organization in order to validate an alert. Frequently, this process of eliminating false alarms includes manually connecting to the resources identified in an alert and analyzing the data found within that resource. In a large or complex enterprise environment this may also require a review of change control records to ensure that a triggered alert itself was not simply a false positive. In the end, most of the validation efforts are driven by the choices and decisions of the monitoring team that may change from time to time. All of these decisions are driven by the need to answer a relatively simple series of questions based on the specific alert that may be detailed and hard to understand. The information required to answer these questions is frequently not captured by the monitoring solution or is contained in disparate locations. To this end, validation of an alert is driven by facts not readily available to an analyst and highly dependent on the staff involved. Moreover, the questions that drive the data collection are rarely documented if at all and security analysts operate based on their experience and intuition, leading to inconsistent results across different operators and large amounts of repetitive manual tasks due to lack of automation.

To improve both the confidence and efficiency in the investigative efforts and reduce the required costs of both time and money we propose to model the requirements for alert validation and automate the collection of required information as much as possible. This methodology is based on the idea that the alert validation process involves a logical model that can be represented as a decision tree resulting in a series of facts constructed in such a way that the outcome obtained gives the analyst a high level of confidence in the final conclusion of whether to escalate the findings as a confirmed security incident or to rule the alert as a false positive. This decision tree requires information that the alert details may or may not contain. For those pieces of information that reside on network resources or collaboration platforms it is likely that collection need to be automated and performed as the alert is presented to the monitoring solution. This collection and caching of information takes the burden of connecting to remote resources and gathering associated data out of the hands of the analyst, reducing the potential for delay in analysis. Moreover, the investigation model also allows for baseline reasoning that guides the data collection process, which increases the consistency and thoroughness of each investigation by modeling and re-using the essential rationale behind investigative reasoning, instead of requiring the analyst to repetitively perform the low-level reasoning to decide the next steps which may vary from one investigation scenario to another. Once collected, the information may be prepared for visualization based on the severity and likelihood of the possible malicious activities. The goal of the presentation is to place the information in front of the analyst in a way that establishes context, enforces relevance to the alert in question and reduces the overall skill and time required to interpret the data.

Part of this work was done while S. Raj Rajagopalan was with HP.

II. A Motivating Example

We use a forensics challenge published by the Honeynet project as a motivating example [3]. In summary, in a pre-planned but unannounced scenario an adversarial role player exploited a vulnerability [1] in the Exim mail relay service. The attacker then used a script to escalate his privilege to root by exploiting another Exim vulnerability [2], then opened a reverse shell and sent the copy of a local disk to a remote server. The forensic challenge provided the disk image and memory dumps from the victim machine. The participants' task was to figure out what happened and answer a number of questions regarding the cause of the attack, its scope, and likely attackers. The solutions from five winners were provided.

While the five solutions varied in the level of detail about the thought processes that produced the answers, it is remarkable that all of them seemed to follow the same investigation steps, which reached the same conclusions. We explain the key findings reported by these solutions, and try to extract from them the ideas for a forensic investigation model that could be used to automate some of the investigative tasks.

1) *What triggered the investigation?* There could be two possibilities each of which is mentioned in some of the solutions. One possibility is that a number of failed SSH login attempts shown in the authentication logs indicate attacks targeting the server. The other possibility is that an error message produced by the Exim attack was written to the panic log which could have triggered the alert. No matter what indeed triggered the investigation, in reality it is typically the case that some anomaly raises suspicion which serves as a critical starting point , from which a series of investigative steps are taken to either validate or refute the alert.
2) *What attacks happened?* All winners of the challenge identified attack strings shown in Exim's main log. What is not completely clear though is why they decided to look at the log in the first place[1]. Depending on what triggered the investigation, two theories could hold:
    a) If the SSH failed login attempts triggered the alerts, the first thing one would do is to examine the authentication log to see if any logins from the offending IP address succeeded. No such entry could be found. However, the failed login is for a user account that did not exist on the system. This is highly suspicious since it could mean that somebody was brute-forcing, or somebody tried to create a new account and use that account to log in (which was the case in this incident as shown by other logs). In either case, one can believe that someone is trying to break into the server. One possibility to move the investigation forward at this point is to scan the machine to identify existence of known vulnerable applications. If this were done then the Exim vulnerability would have been reported, and the investigation will move on to the Exim log. Alternatively, one could look at listening ports on the server and there are only a few services running on server, which is a simple task considering that Exim being one of them. The investigator can then decide to look at the logs of all the services and would have uncovered the attack strings in the Exim log.
    b) If it were the Exim panic log that triggered the investigation, then the logical next step would be to look at the Exim main log. In this case it would be much more straightforward to locate the attack strings.
3) *What is the scope and impact of the attacks?* All five solutions agreed on what the attacker tried to do and how successful he was. For this incident, once the attack strings on the Exim main log are found, it is likely a quick and easy process to understand the full scope of the attack, because the attacker does not seem to have tried hard to cover his tracks and left several indicators of activity.

[1]There were dozens of processes running on the machine.

Observations from this forensic challenge

The five solutions for this forensic challenge indicate that there is some coherent and perhaps subconscious logical process that is being applied when a security analyst investigates an incident. The key part is how to connect the dots and go quickly from one piece of data to another until the full scope of the incident becomes clear. When the most relevant data pieces are all presented, it is not hard to see what and how events are interlinked. The difficult part is finding those pieces. The goal of our research is to design a model for the process of forensic investigation, with an eye towards providing a high degree of automation to aid an analyst to find the most relevant information.

III. Investigation Modeling

Based on one of the authors' experience in providing second and third tier support for IDS alert investigations over the last nine years it has been observed that while an alert may be the outcome of the monitoring process it is frequently just the beginning of the real investigation. As such a computer security team finds itself in the unenviable position of taking the alert details and applying some logic to them in order to make the determination as to whether the alert is a confirmed security incident or a false positive. This process is required for each alert that the business deems important (classified as *critical*). This set of critical alerts form the basis for the investigative model. For each of these alerts (or a robust subset as determined by historical data), the facts required to validate the alert are pre-documented. Each fact is then itself classified in accordance with its *source, required finding, and significance*. The sources are the application logs, audit records,

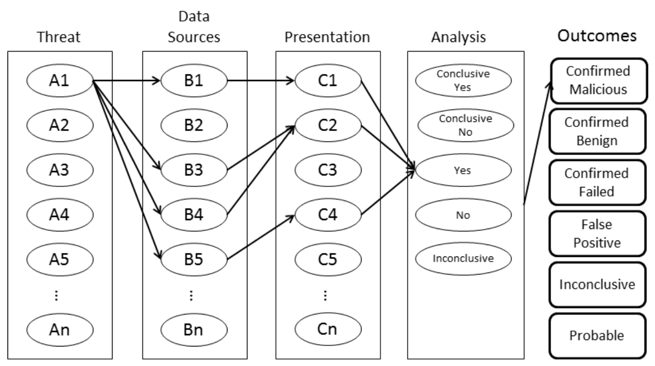

Fig. 1. Threat Modeling and Predictive Data Collection (Scenario 1)

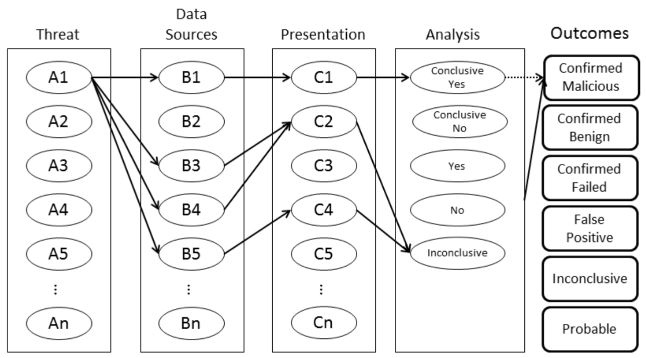

Fig. 2. Threat Modeling and Predictive Data Collection (Scenario 2)

and any other object containing the required information. The required finding is the result of analysis that is used to establish a given fact. The significance is a ranking of the fact in its ability to classify the alert as a true or false positive. This significance can be established in many ways and may be subject to modification based on available resources or specific features of an organization's information systems.

Table I shows some critical alerts and facts needed to confirm or rule out the alert. Those facts are established through examining additional data sources to answer the relevant questions. The "Required Finding" indicates the facts that would rule out the alerts and these facts have varying degrees of significance explained in the table.

In our research we would like to formally specify such criteria in a logical model, with certainty qualifiers indicating in a generic manner the significance of the various facts. Fig. 1 - 3 show some possible scenarios that this logic may play out in an investigation process.

In Fig. 1, all the data sources indicate that the alert is caused by malicious activities (Yes in the Analysis column). The final outcome is "Confirmed Malicious". In Fig. 2, one data source provides conclusive evidence that the event is malicious (dotted line). In Fig. 3, some data indicate the alert is malicious while another indicates it as false positive. The final outcome is a "Probable". We have performed research on methods to formally specify the qualifiers such as "conclusive yes", "yes", and "inconclusive" and use them to model this reasoning process. The *significance* weight in Table I could be translated into such qualifiers. How to fuse the evidence with varying shades of uncertainty will be an important topic and we will leverage our prior work in reasoning under uncertainty for cybersecurity to tackle this challenge [7, 11].

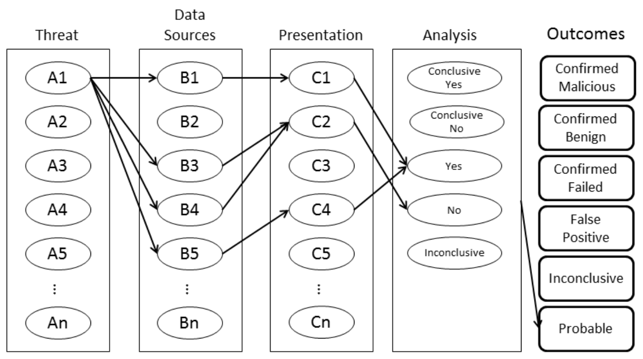

Fig. 3. Threat Modeling and Predictive Data Collection (Scenario 3)

IV. A Forensic Analysis Tool

We will center our research around a forensic analysis tool that will provide some baseline reasoning and predictive data collection. Fig. 4 illustrates our vision of the tool. The user interface of the tool consists of a Threat Analysis Console that provides a prioritized view of all the data relevant to the current analysis. A human analyst interacts with the console to move the investigation forward. On the lefthand side we show an example alert (SSH brute-force attempt) which triggers the investigation. The back-end engine of the forensic tool will automatically extract the relevant segments of various logs and filter out entries of interest to display in the console.

The analyst will apply the analysis via the dialogue buttons to the right of each "Fact" window. Once each item has an entered value our investigation model kicks in and processes the new inputs to determine the state of the alert/attack. With the analysis of the collected information supplied by the analyst via the console the model will return with a guided response depending on the determined outcome. As the research progresses we would like to see the possibility of further automating the process by allowing the model to automatically assign the various significance weights to the facts (as opposed to requiring the user to press the various buttons for every fact), further reducing the labor required from a human analyst to guide the reasoning process. The ultimate goal of the research is to yield a forensics assistance tool that will require human input only when the analysis requires a high-level of intelligence not replicable by a computer program.

Alert	Examples		
	SSH brute-force attempt.	SQL Injection.	User account added to Administrator's group.
Question	Did attack source IP address successfully connect to server?	Did web server provide anything but non-descriptive error pages to suspect IP address?	Was change to user account authorized?
Source	Authentication log on target host.	Web server error log on target host.	Identity management change control record.
Required Finding	All authorization events from source IP address must be failed or blocked attempts.	All web requests from source IP address were denied with standard 404, 405 or non-descriptive 500 error codes.	Specific account was approved for noted change on date observed.
Significance (out of 10)	7	8	10
Reasoning for ranking	Observing a lack of authorized connections from the source IP address indicates failure to brute force an account but does not preclude success by other attack strategies and therefore cannot provide conclusive evidence.	No successful page returns (status 200s or 300s) indicate that the web server correctly refused invalid URL requests.	An approved change request for user account modification should be the definitive record of an authorized event.

TABLE I
INVESTIGATION MODELING

A. *Information Presentation*

It is highly critical that the tool presents the information in a manner that facilitates human analysis. The console will have functionalities such as text highlighting and manipulation, and alignment of the data so that hidden causality can be more easily caught. It also needs to provide user interaction needed to process the information and move forward the investigation. For example, in the first window the user would be asked to confirm that all traffic from the intruder's IP address was denied. The user would scroll through the events or "grep out" the failed attempts and make the assessment. If all activity was "denied", the user would click the 'No' option. The user would be guided through each window and when complete they would submit the findings for analysis. If the model can determine the status of the event it will make the call without waiting for the user to make a decision. For example, if the console shows a change-control record allowing the action and the user says "yes" it was approved, the model should make the call that this is not a problem and close out the alert. If this type of reasoning is a common pattern we could include it in the model so that the user does not need to do anything in such situations.

V. RESEARCH IN PROGRESS

We are currently developing the investigative model based on a large number of scenarios from real incidents. We are also building the forensic analysis tool that applies the model in the back-end engine and display/prioritize the data entries from the log files as illustrated in Fig. 4. We intend to deploy the tool to be used by a number of security analysts and observe the tool's effectiveness as we build it.

VI. RELATED WORK

Case et al. [4] developed a correlation model for forensic analysis. The model can process data from memory dumps, network traces, disk images, log files, user accounting and configuration files. Data from all the sources are loaded into a web interface and the events are manually correlated. Hayoz et al. [5] have developed a model that enables an analyst to do forensic analysis visually. Data from multiple network and host entries are read using existing forensic analysis tools and then placed into abstract classes. The analyst then visually constructs the correlation scenario by querying for more data within each class. Once the analyst constructs a successful scenario that sequence can be stored for future use. Peisert et al. [9] propose a forensic data collection framework based on structure of attacks. Their idea is that by modeling the various attacks one can extract the logging requirements and hence

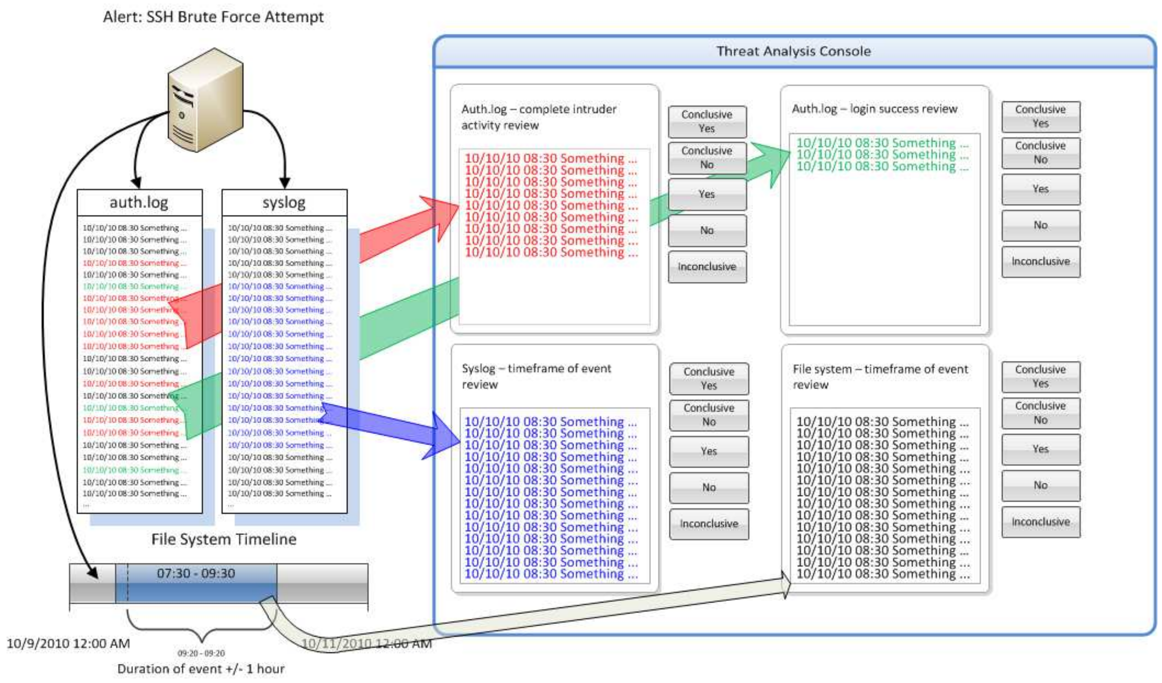
Alert: SSH Brute Force Attempt
auth.log
syslog
File System Timeline
07:30 - 09:30
10/9/2010 12:00 AM
10/11/2010 12:00 AM
Duration of event +/- 1 hour
Threat Analysis Console
Auth.log – complete intruder activity review
Auth.log – login success review
Syslog – timeframe of event review
File system – timeframe of event review
Conclusive Yes
Conclusive No
Yes
No
Inconclusive

Fig. 4. Vision of the Forensic Analysis Tool

when those attacks happen in real-time you have the data to do the forensic analysis.

Rekhis et al. [10] developed a language using which attacker activity can be modeled as state transitions. The idea is to start from the initial non-compromised state and advance to the next probable state based on the available evidence. One can hypothesize for missing evidence and also avoid impossible states by using conflicting evidences during this process. One drawback though is that the model must be abstract enough to be usable across multiple scenarios otherwise it has to be customized for each investigation. The interesting aspect of this model is that each rule in the model contains information on where to look for evidence, close to what we are trying to achieve. King et al. [6] developed a tool called BackTracker that works by observing files, filenames, processes and system calls. Given a detection point, the tool works backwards by building a dependency graph consisting of the mentioned operating system elements. Peisert et al. [8] in another work developed an anomaly detector that can detect malicious code execution by looking at sequence of function calls made by that program.

Each of the prior work mentioned above focuses on a single aspect of forensics, such as efficient data logging or correlation of evidence. We propose an approach that models multiple components of forensic reasoning, with an eye towards predicting and automating the data collection process for a given event. Our method needs to deal with the inherent uncertainty in the analysis process which has not been addressed by prior works. We base our reasoning model on concrete experience from security analysts working in the trench, and apply an iterative/spiral approach to tool building so that quick feedback from users can be incorporated into the model design and tool building continuously.

Current commercial security monitoring consoles collect, aggregate and correlate a significant amount of data. The analysis that an end user can perform consists of drilling down through the various features of the alerts they are presented with. Beyond the contents of the events and the resultant alerts these consoles offer little assistance in the way of investigation support. Our methodology not only understands the information required to validate an alert but also proactively collects that information, displays it to the user in a guided presentation and provides direction based on the user's response. This approach moves the security monitoring console one step closer to being a comprehensive solution instead of a simple means of display.

VII. Acknowledgment

This material is based upon work supported by U.S. National Science Foundation under award no. 0954138, and HP Labs Innovation Research Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, or Hewlett-Packard Development Company, L.P.

References

[1] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-4344.

[2] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-4345.

[3] Challenge 7 of the forensic challenge 2011 - forensic analysis of a compromised server. http://www.honeynet.org/challenges/2011_7_compromised_server.

[4] Andrew Case, Andrew Cristina, Lodovico Marziale, Golden G. Richard, and Vassil Roussev. Face: Automated digital evidence discovery and correlation. *Digital Investigation*, 5, Supplement(0):S65 – S75, 2008. The Proceedings of the Eighth Annual DFRWS Conference.

[5] M. Hayoz and U. Ultes-Nitsche. Visual correlation in the context of post-mortem analysis.

[6] Samuel T. King and Peter M. Chen. Backtracking intrusions. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 223–236, New York, NY, USA, 2003. ACM.

[7] Xinming Ou, S. Raj Rajagopalan, and Sakthiyuvaraja Sakthivelmurugan. An empirical approach to modeling uncertainty in intrusion analysis. In *Annual Computer Security Applications Conference (ACSAC)*, Dec 2009.

[8] S. Peisert, M. Bishop, S. Karin, and K. Marzullo. Analysis of computer intrusions using sequences of function calls. *Dependable and Secure Computing, IEEE Transactions on*, 4(2):137 –150, april-june 2007.

[9] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. Toward models for forensic analysis. In *Systematic Approaches to Digital Forensic Engineering, 2007. SADFE 2007. Second International Workshop on*, pages 3 –15, april 2007.

[10] Slim Rekhis and Noureddine Boudriga. A temporal logic-based model for forensic investigation in networked system security. In *Proceedings of the Third international conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, MMM-ACNS'05, pages 325–338, Berlin, Heidelberg, 2005. Springer-Verlag.

[11] Loai Zomlot, Sathya Chandran Sundaramurthy, Kui Luo, Xinming Ou, and S. Raj Rajagopalan. Prioritizing intrusion analysis using Dempster-Shafer theory. In *4TH ACM Workshop on Artificial Intelligence and Security (AISec)*, October 2011.