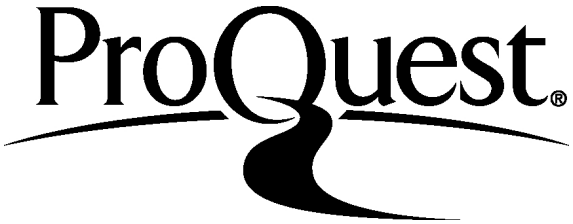
HUMAN-CENTRIC TRAINING AND ASSESSMENT FOR CYBER SITUATION AWARENESS

by

Zequn Huang

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Sciences

Fall 2015



HUMAN-CENTRIC TRAINING AND ASSESSMENT FOR CYBER SITUATION AWARENESS

by

Zequn Huang

Approved: _____
Kathleen F. McCoy, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: _____
Babatunde Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: _____
Ann L. Ardis, Ph.D.
Interim Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Chien-Chung Shen, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Adarsh Sethi, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

John Cavazos, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

John D'Arcy, Ph.D.
Member of dissertation committee

To my parents.

ACKNOWLEDGEMENTS

Foremost, I would like to thank my advisor, Dr. Chien-Chung Shen, for his support, encouragement and inspiration. Over the past years, Dr. Shen has been providing so many resources and help to accommodate my research and study. Also, Dr. Shen has been constantly encouraging me to delve further into the problems at hand, as well as to keep an open mind on future research topics. Moreover, Dr. Shen's enthusiasm for research and his dedication to work have deeply influenced me since my first day at the University of Delaware. Without the mentorship and help from Dr. Shen, my Ph.D. journey could have been longer and harder.

I am also grateful to my committee members, Dr. Adarsh Sethi, Dr. John Cavazos and Dr. John D'Arcy. Dr. Adarsh Sethi helped me with several thesis writing problems and provided precious suggestions. Dr. John Cavazos and made valuable comments on my proposal and dissertation. Dr. John D'Arcy provides valuable opinions about thesis topic, focusing not only on the Cognitive Task Analysis issues but also on the challenges raised from Team Cyber Situation Awareness.

I feel very fortunate to work with a group of talented research colleagues, Dr. Ke Li, Dr. Yang Guan, and Dr. Rui Fang. We have been sharing knowledge together and helping each other with research projects. It was my privilege of working with all of them. I also greatly appreciate the assistance of our department staff, Teresa Louise, Krystal Proctor, Samantha Fowle and T. Gregory Lynch, throughout my study at the University of Delaware.

Finally, I want to thank my parents for making many sacrifices that allowed me to get to this point. They have respected every decision I made, been happy for every little thing I achieved and supported me in every way they could. I could not have gone through the process without the support from my parents.I thank my parents for being here for me, for sharing my happiness and sadness, for being proud of every achievement I have made and

for all the sacrifices you made for our family. Your love will always be the motivation of my work and life.

TABLE OF CONTENTS

Appendix

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Cyber attacks have been increasing significantly in both number and complexity, prompting the need for better training of cyber defense analysts. One issue with existing cyber security training is that it relies mostly on lecture-style instructions without much hand-on experience. Thus, need a training solution that provides a realistic, human-in-the-loop environment for cyber analysts to explore, collaborate, and interact for effective learning.

Situation Awareness (SA) means the comprehension and perception of environmental elements with respect to time and space. Cyber Situation Awareness (CSA) is SA extended to the cyber domain. During CSA, the cyber analysts need to understand the meaning of the observations and be able to project the impact of the observations to the system. In this proposal, we describe a Cyber Situation Awareness training and assessment system with the purpose of teaching and measuring individual and team cyber situation awareness within the cyber defense context, as well as incorporating various technologies to enhance the cyber analysts' learning process.

To conduct effective cyber security training, it becomes essential to design realistic exercise lesson plans. Accurate identification of experts' cognitive processes through Cognitive Task Analysis can be adapted into training materials to teach novice cyber analysts (or trainees in this proposal) how to think and act like an expert during defense. In order to solve the information overload challenge faced by trainees, we identify and design watch list statistics, which allows trainees to tailor their own watch list statistics and triggering threshold conditions in order to recognize cyber attacks faster. The speed with which a trainee can recognize, analyze, and respond to attacks is critical as it will limit the damage and lower the cost of recovery. Therefore, we evaluate trainees' performance based on their response time comparing with estimated attack ground truth timeline. We also devise scoring algorithms

to calculate trainees' performance scores according to the weighted functions combing all performance metrics.

Then, as training is an iterative process, the assessment component not only assesses the knowledge gained by the cyber analysts, but also adjusts the difficulty of training lessons accordingly based on trainees' performance. Nevertheless, quantifying difficulty level of training lesson scenario is an important but difficult task. While standard techniques exist for measuring the relative difficulty to exploit individual vulnerability, it is challenging to answer the fundamental question whether one scenario containing several vulnerabilities is more difficult than another one. Based on causal relationships between vulnerabilities in attack graph, we apply Bayesian Reasoning to aggregate individual vulnerabilities into a probabilistic value representing the attackers' success likelihood to achieve the attack goal. Based on the quantified probability of achieving attack goal, the lesson's difficulty-level is categorized accordingly.

Furthermore, complex and dynamically changing task such as cyber defense often requires the effective coordination of a team of cyber analysts. Cyber analysts need to work collaboratively as a team at different levels and different parts of system. Each team member collects data, generates its own awareness for the cyber situation, and shares with other team members to get the comprehensive understanding of the overall situation for the decision making purpose. Since each team member may have his/her own personal expertise knowledge, experience, and opinions, it is hard for the entire team to make consensus decision when having conflicting judgments. Considering human cyber analysts tend to use ambiguous linguistic language to express their own cyber situation awareness during the team discussion process, we design a fuzzy set based method to facilitate cyber analysts to quantify their thoughts and make consensus decision that is most acceptable by the entire team.

Finally, we investigate real-time purpose sensors based information fusion for Advanced Persistent Threats (APTs) detection. As human cyber analysts have to examine huge amount of data such as system logs, configuration files, traffic logs, IDS log, and audit logs in order to identify potential threats. Thus, they would be soon overwhelmed by tremendous

data and forced to ignore potentially significant evidences introducing errors in the detection process. Furthermore, cyber security attack and anomaly detection techniques suffer from their reliance on known malicious signatures or unusual conditions that warrant further investigation. The use of signature-based detection cannot effectively eliminate false negatives when dealing with Advanced Persistent Threats, since the financial resources and time available to APTs allows the use of previously unknown 'zero-day' attack vectors. The designed information fusion system reduces an operator burden to handle false positives and reduces time to detection while using noisy/high false positive inputs. We design and develop host- and network- based purpose sensors and places them within the network and individual hosts to provide real-time purpose and correlation inputs and then use this information combined with network-specific knowledge to create a dynamic set of event threads that, when touched by a given alert received from traditional intrusion detection systems (IDSs) such as Snort and OSSEC, allows the immediate identification of the context surrounding the alert and thus the automatic calculation of the alert's legitimacy and severity.

Chapter 1

INTRODUCTION

1.1 Background and Motivation

Cyber attacks, which refer to any computer-to-computer attacks that undermine the confidentiality, integrity, or availability of computer or information resident on it, have increased significantly in both number and complexity in recent years. Typically, a cyber attacker first exploits a system's vulnerabilities and infiltrates its network and/or hosts. Once the attacker gained entrance into the system, he/she may use it to monitor communications, steal critical data, discover new avenues of attack in related systems, take control of assets managed by the system or disable networks, computers, and associated systems. Harmful outcomes of a successful attack include the attacker's accessing sensitive data on the network and controlling the hosts and network resources.

Situational Awareness (SA) involves perception of evolving status and attributes of elements, comprehension of combined observations to evaluate the current situation, and prediction of possible future outcomes based on past experience and knowledge [47]. Cyber Situation Awareness (CSA) extends SA to the cyber domain, where cyber analysts collect data and seek cues to form attack tracks, estimate the impact of observed attack tracks, and anticipate the moves (actions, targets, time) of attackers. However, presently effective CSA is hampered by the enormous size and complexity of the network, by the adaptive nature of intelligent adversaries, by the high number of false alarms generated by Intrusion Detection Systems (IDSs), by the lack of ground truth to assess defense performance, by organizational stove-pipes thwarting collaboration, and by technologies that lack an adequate understanding of the human needs.

In particular, in contrast to environments that are bounded by physical constraints and/or geographical features, cyberspace possesses the following unique features, which

further impose extraordinary cognitive challenges on cyber analysts. First, while a cyber analyst is fully aware of the boundaries of his/her managed networks, the external cyberspace is boundless with minimal geographical features. As a result, the environment from which a cyber analyst has to perceive salient cues is vastly larger and more difficult to comprehend. Comprehending even a small segment of cyberspace is challenging. Second, the speed at which the cyberspace changes is much faster, where new vulnerabilities and their corresponding exploits are continuously emerging, and new offensive technologies are constantly being developed. Furthermore, modern exploits are either employed via misdirection (e.g., a DDoS attack is conducted by a botnet of compromised computers) or delivered passively via embedded malware. Third, everything a cyber analyst knows about the environment is a virtual representation of the cyberspace in terms of digital information (e.g., intrusion alerts and firewall logs). In addition, the cyber analyst only sees the information that his/her (software) sensors are capable of detecting in a form that can be rendered on a monitor screen. Because perception and comprehension of cyberspace is inherently constrained by technology artifacts, cyber analysts' ability to develop situation awareness is greatly limited by the degree to which the network's sensors are correctly configured and capturing data.

Furthermore, cyber analysts are faced with extraordinary amounts of information (such as various IDS and audit logs) to sift through, and CSA demands that various pieces of information be connected in both space and time. This connection necessitates team collaboration among cyber analysts working at different levels and/or on different parts of the system. It is anticipated that team CSA can be carried out to systematize information coordination and team collaboration for CSA effectiveness and resilience. As cyber attacks are becoming more frequent and more complex, the need for more effective training of cyber analysts and their collaborative efforts to protect critical assets and ensure system security become more crucial and urgent. In this thesis, we address the following three closely related topics.

1.2 Effective Cyber Situation Awareness Training Methodology

The dramatically increasing number and complexity of cyber attacks prompts the need for more effective training of cyber analysts. However, most existing cyber security training relies heavily on lecture-style instructions without enough hand-on experience. Cyber security training could benefit significantly from empirical scenarios to let trainees practice deciphering ever-sophisticated attacks. Given that cyber security defense is a cognitive process for human cyber analysts, we design training solutions that utilize the approach of Cognitive Task Analysis (CTA).

CTA is the extension of traditional task analysis techniques to yield information about the knowledge, thought processes and goal structures that underlie observable task performance. The outcome of CTA describes the performance objectives, equipment, conceptual knowledge, procedural knowledge and performance standards used by experts as they perform a task. Accurate identification of cyber security experts' cognitive processes can be adapted into training materials to teach novices how to perform like experts.

We adopt a CTA-based approach to gain insight into the cognitive demands and workflow of cyber analysts and design cyber security training lesson plans and training workflow. We then evaluate cyber analysts' performance during training and adjust the training scenarios to first accommodate and then further improve their skills.

To conduct effective cyber security training, it becomes necessary to quantify the "difficulty level" of different attack scenarios to accommodate a diverse group of trainees with different skill levels. In the beginning, a trainee is given attack scenarios of appropriate difficulty levels. As trainees improve their skills in cyber security training, more difficult scenarios could be given in succeeding training. While standard techniques exist for measuring the relative difficulty of exploiting individual vulnerabilities, it is challenging to answer the question of "whether one attack scenario involving multiple vulnerabilities is more difficult than another attack scenario." Naive ways of aggregating the difficulty levels of individual vulnerabilities, such as taking the average or maximum values, may lead to misleading results.

In particular, we design a probabilistic metric to quantify the difficulty levels of attack scenarios. Specifically, we apply Bayesian Reasoning to an adapted attack graph of an attack scenario to aggregate the CVSS Exploitability sub scores of individual vulnerabilities involved in the attack scenario into a probabilistic value representing the likelihood of a successful attack. This aggregation is based on the causal relationships that exist between vulnerabilities in an attack graph. However, one major constraint of using Bayesian Reasoning is that it does not work with cycles, which are common in attack graphs. To address this issue, we identify different types of cycles in attack graphs and design an efficient algorithm to remove cycles while preserving cyclic causality in the probability calculation. We use empirical scenarios to evaluate our approach and present one case study.

1.3 Team Collaboration for Cyber Situation Awareness

Complex and dynamically changing task such as cyber defense often requires the effective coordination of a team of cyber analysts. Cyber analysts need to work collaboratively as a team at different levels and different parts. Each team member collects data, generates its own awareness for the situation, and shares with other members to get the comprehensive understanding of the overall situation. Previous researches conclude the observation from the Air Force Academy's participation in the Cyber Defense Exercise (CDX) as the team of cyber analysts does not work as expected. One major issue is that there is a lack of clear team structure and role assignment, resulting in such a situation that an analyst would not know who to ask for help and who is responsible for what. Another issue is that cyber analysts tend to work independently with no communication. Analysts reported that they occasionally worked on the same data set as other analysts, and only discovered this after the task was completed.

The most critical issue for team collaboration is that each team member may have his/her own personal expertise knowledge, experience, and opinions, which make it hard for the whole team to make consensus decision when having conflicting judgments. Traditional methods to achieve consensus decision such as through verbal discussion and whiteboard session are not accurate and unpersuasive. Therefore, there is a need for a quantization

team decision-making technique to select the satisfactory solution that is most acceptable for entire team of cyber analysts.

We utilize fuzzy set to let team members achieve consensus decision for the situation. Fuzzy set has been well applied in the area of multi-criteria team decision-making to deal with uncertain issues in generating consensus opinions. It can construct preference relation between alternatives by evaluating different criteria, and select the best action from a set of alternatives that is most acceptable by the entire team. In particular, we use fuzzy set to facilitate a team of cyber analysts to make consensus decision when they have conflicting judgements on the type of on-going attacks.

1.4 Real-time Information Fusion for Cyber Situation Awareness

From a cyber analyst's perspective, there are several cyber security tools available. Each of the tools generates various types of data about the current network status. Log data from servers and hosts provide another incomplete picture of the current cyber situation. Connection or flow data from Netflow, for instance, can only provide the communication information between systems. Cyber analysts have to dig deeper into tremendous amount of data manually to identify anomaly behavior. Besides, the data tends to have high false alert rates. Thus, a better understanding needs to be developed through a suite of tools that provides analysts with the necessary and sufficient information to maintain awareness of the events that occur on the network. Unfortunately, the state-of-the-art cyber-security maintains these tools separately, which forces cyber analysts to select and use them individually.

To address this critical need of enabling analysts to rapidly and accurately detect Advanced Persistent Threats (APTs), we design a novel non-signature based APTs detection system that allows the proper identification, prioritization, and understanding of APTs attacks. The key innovation of our approach is to design and develop host-and network based purpose sensors and places them within the network and individual hosts to provide real-time data and then use this information combined with network-specific knowledge to create a dynamic set of event threads that, when touched by a given alert received from traditional Intrusion Detection Systems (IDSs), will immediately identify the context surrounding the

alert and thus the automatic calculation of the alert's legitimacy and severity. The result is that much of the follow-up investigation of each alert is shifted into the prioritization process that utilizes the gathered context for correctly prioritizing the analyzed alerts. The burden on cyber analyst is largely reduced both by significantly improved prioritization and by providing a contextual picture of each identified potential attack. Hence, APTs attacks can potentially be detected while still at a preparation stage and with an operationally relevant level of accuracy.

1.5 Thesis Outline

The goal of the thesis work is to develop a training and assessment system for individual and team cyber situation awareness. We have designed: (1) Live-Virtual-Constructive based Cyber Situation Awareness Training and Assessment Framework; (2) Cognitive Task Analysis based Lesson Plans for Cyber Situation Awareness Training and Assessment; (3) Difficulty-Level Metric for Cyber Security Training and Assessment System; (4) Team Collaboration for Cyber Situational Awareness; and future research objective on (5) Realtime Context-awareness based Information Fusion for Cyber Situation Awareness.

In Chapter 2, we first review related work. Besides teaching cyber security through lecture-style instructions, several projects have pioneered the use of cyber defense competitions as teaching method of learning through practice, such as Cyber Defense Exercise, Capture The Flag competitions, and so forth.

In Chapter 3, Live-Virtual-Constructive (LVC) framework is introduced. LVC framework has the advantages of combining real equipment running real applications, real people operating on simulated systems, and simulated people operating on simulated systems. Based on this framework, we design a cyber security training and assessment system that delivers realistic hands-on exercises that enable trainees to experience cyber attacks in wired and wireless networked environments, and to learn how various procedures and techniques are more effective or less effective at recognizing and countering those attacks.

In Chapter 4, the technique of Cognitive Task Analysis is used to capture and represent knowledge used by experts to perform complex tasks. Accurate identification of experts'

cognitive processes can be adapted into training materials to teach novice cyber analysts how to think and act like a security expert. After performing Cognitive Task Analysis, we identify the steps necessary for designing cyber security lesson plans for cyber security training system. Using these steps, we design three cyber security training lesson plans: port/network scanning, denial of service, and wireless jamming. In these lesson plans, we specify the lesson scenarios and design watch lists that cyber analysts should observe to detect ongoing attacks. We also devise scoring algorithms to evaluate cyber analysts' performance according to weighted functions of performance metrics.

In Chapter 5, we describe difficulty-level metric for cyber security training System. As cyber security training is an iterative process, the assessment component not only assesses the knowledge gained by the cyber analysts, but also adjusts the difficulty of training lessons accordingly based on the analysts' performance. In this chapter, we present an attack graph-based probabilistic metric to measure lesson scenarios' difficulty levels. Based on causal relationships between vulnerabilities in an attack graph, we apply Bayesian Reasoning to aggregate individual vulnerabilities into an probabilistic value representing the attackers success likelihood to achieve the attack goal. However, one major complication of using Bayesian Reasoning is that it does not allow for cycles, which exists in attack graphs. We identify different types of cycles in attack graphs and propose an efficient algorithm to remove cycles while keeping cyclic influence in the probability calculation.

In Chapter 6, a quantization method is described to resolve a team of cyber analysts' conflicting judgements and make consensus decision. Faced with extraordinary amounts of information and dynamically changing environment, cyber analysts need to work collaboratively as a team. Each team member collects data, generates the awareness for the situation, and shares with other members to get the comprehensive understanding of the overall situation for decision making. Nevertheless, each individual may have his/her own personal expertise knowledge, experience, and opinions. It is hard for the whole team to make consensus decision when having conflicting judgments. Considering that human cyber analysts tend to use ambiguous linguistic language to express their own cyber situation awareness during team discussion, we describe a fuzzy set based method that allows cyber analysts

to quantify their preference and make consensus decision on the cyber attack types that are most acceptable by the entire team.

Most of the existing cyber security attacks and anomaly detection techniques suffer from their reliance upon known malicious signatures or unusual conditions. However, the use of signature based detection technique cannot effectively eliminate false negatives when dealing with Advanced Persistent Threats. In Chapter 7, we approach the problem by deploying a "wide net" of network, host, and purpose sensors to collect the context surrounding the alerts being generated at different levels throughout a network. In order to minimize the burden on a cyber analyst's limited capacity to correctly prioritize and act upon these alerts, all alerts and sensor data are collected and stored in database and are then processed, correlated, and prioritized into meaningful events that are presented to the analyst in the order of importance. The designed APTs detection system is capable of effectively generating appropriate event contexts for each attack type and using this context to discern the legitimacy of an alert type by calculating its similarity with the APTs profiles.

Finally, Chapter 8 summarizes the contributions of the dissertation and describes future research directions.

Chapter 2

RELATED WORK

Besides teaching cyber security through lecture-style instructions, several projects have pioneered the use of cyber defense competitions as a teaching method which emphasizes learning through practice. Cyber Defense Exercise (CDX [84, 4]) is an advanced cyber skill building exercise that focuses on network security from a defensive point of view. The exercise consists of several participants including the students who operate and secure networks, the red team acting in the role of attackers who attempt to exploit known vulnerabilities, and the white team acting as the exercise controllers and score keepers. Motivated by the competitive sprit of CDX, students can enhance their learning experiences.

Capture The Flag (CTF [100, 24]) competitions are also great tools to teach students hands-on lessons about cyber security. Different from CDX which focuses on defense, CTF also hones offensive techniques. The class was divided into two teams and both teams should protect a set of hosts and hide a flag such as a secret file. In addition, each team had to attack the other teams hosts and retrieve the flags for each of the attacked hosts. Hansen [33] proposes a realistic cyberspace-training environment called Cyber Flag. It combines the best practices of existing training (both military and civilian) with the most realistic cyberspace threats and scenarios.

General reviews of current simulation-based security training systems are given in [74]. CyberProtect [34], developed by US DoDs Defense Information Systems Agency (DISA), is developed as a training tool for novice network security analysts to familiarize them with system security terminology, concepts, and policy. Through interactive security defensive exercises, the trainees make practical decisions for allocating resources after risk analysis and risk management. Duffy [23] proposes a network defense training system through CyberOps network simulations. It is similar to an interactive video game in which the analysts

select the security defensive tools and solutions and then attack sequences generated by computer that are launched to assess the effectiveness of the selected solutions.

The US Military Academy designs the Military Academy Attack/Defense Network (MAADNET [90]) learning environment. This application is built on a client-server architecture using the discrete events simulation paradigm. The exercise network can be constructed with different network components such as switches, routers, workstations, wireless access points, etc. Each of these components has one or more traffic generators associated with it. The users can build their own networks, adopt polices, and employ different types of administrative iterations. Then, several attacks can be performed against the designed network to see how well it performs during attacks. After the simulation, the built network is evaluated in order to assess how well security was maintained.

CyberCIEGE [44] is a game-based cyber security teaching and learning environment. Inspired by the success of games such as SimCity and RollerCoaster Tycoon at capturing users attention, CyberCIEGE provides a similar game-style simulator for security training purposes. The students are put in the position of the decision maker of an IT organization. The aim of the game is to protect the system by using appropriate security measures involving procedures, physical and technical security, while keeping the virtual users productive and pleased. CyberCIEGE has the advantage of extensibility that allows educators to develop scenarios tailored to their classes, while the open sharing model allows educators to share and reuse scenarios.

A Reconfigurable Attack-Defend Instructional Computing Laboratory (RADICL [11]) aims to provide a flexible alterable laboratory environment. The objective is to prepare students to understand attack scripts and other malware and to use defensive strategies and tools. It is not based on the simulator environment. Instead, workstations are set up to enable students to switch operating systems and reconfigure the network topology.

The Real-time Immersive Network Simulation Environment (RINSE [56]) is a highly extensible simulator designed for large-scale, real-time cyber-security training and exercises. It provides multi-resolution traffic modeling, efficient attack models, efficient routing simulation, and CPU/memory resource models for large scale preparedness and training exercise.

It utilizes client-server architecture so that the client application Network Viewer allows the users to monitor and control the simulated network from the client side. From there, the user can issue several commands in order to influence the model behavior. There are five different types of commands: attacks, defenses, diagnostic networking tools, device control, and simulator data.

Network Security Simulator (NeSSi [13]) is an open source discrete event network simulator that contains several security-related capabilities such as profile-based automated attack generation, traffic analysis, and interface support for the plug-in of detection algorithms. NeSSi has the extensibility advantage by adopting a plug-in mechanism. The plug-in concept allows the functionality extension without changing the simulation core itself. These extension mechanisms allow three different levels of abstraction: application, network, model and device level.

R. J. Guild developed the Reconfigurable Cyber-Exercise Laboratory (RCEL [30]). The aim of RCEL is to create a cyber laboratory environment with flexible collection of equipment that can be quickly interconnected and configured. The lab configuration can be quickly changed for different activities. In this case, the author proposes the use of Symantec Ghost to create images of pre-configured stations that could then be rapidly deployed when required. Several prototype cyber-exercise scenarios such as limited interaction attack, external network vulnerability, and aggressive cyber exercise are presented to supplement the RCEL description.

CyberCog [76] is a synthetic task environment for understanding and measuring individual and team situation awareness, and for evaluating algorithms and visualization intended to improve cyber situation awareness. CyberCog provides an interactive environment for conducting human-in-the-loop experiments in which the participants perform the tasks of a cyber analyst in response to a cyber attack scenario. CyberCog generates performance measures and interaction logs for measuring individual and team performance. CyberCog has been used to evaluate team-based situation awareness. CyberCog utilizes a collection of known cyber defense incidents and analysis data to build a synthetic task environment. Alerts and cues are generated based on emulation of real-world analyst knowledge. From

the mix of alerts and cues, trainees must react to identify threats (and vulnerabilities) individually or as a team. The identification of attacks is based on knowledge about the attack alert patterns.

Designed for better understanding of the human factor in a cyber-analysis task, idsNETS [62], built upon the NeoCITIES Experimental Task Simulator (NETS), is a human-in-the-loop platform to study situation awareness for intrusion detection analysts. Similar to CyberCog, NETS is also a synthetic task environment. The realistic scenarios are compressed and written into scaled world definitions and the simulation engine can interpret them in a simulated environment, run the simulation, and respond to user interaction. In [30], several human subjects experiments have been performed using the NETS simulation engine to explore human cognition in simulated cyber-security environments. The study indicates that the teams who had more similar skill sets displayed a more cohesive collaboration via frequent communication and information sharing.

In contrast to most existing cyber security training systems, such as CyberCog and idsNETS that employ a synthetic task environment, we design a human centric training and assessment system that is designed by using the Live-Virtual-Constructive (LVC [94]) framework, which facilitates scalable, programmable, and realistic training and assessment for cyber security. The LVC framework is based on actual simulation of the operational systems. A synthetic task environment may rely on previous incidents to generate the sequence of alerts and cues corresponding to those incidents, LVC framework is able to simulate previous incidents as well as generate new simulated or emulated incidents on the fly. Thus, LVC is useful as a war game rehearsal tool. The LVC framework also supports a hybrid network of actual and virtual machines so that attacks can be launched from an actual or a virtual host, targeting an actual or a virtual host. In essence, an LVC framework provides a real-time hardware-in-the-loop capability for simulation of cyber threats to the entire infrastructure where the impacts of cyber attacks can be tested on actual systems.

Chapter 3

LIVE-VIRTUAL-CONSTRUCTIVE BASED CYBER SITUATION AWARENESS TRAINING AND ASSESSMENT SYSTEM FRAMEWORK

The goal of the proposed work is to develop training and assessment system to be used for teaching and measuring individual and team cyber situation awareness within the cyber defense context. The training system is based on Live-Virtual-Constructive (LVC) framework because it can provide hands-on interactive scenario that realistically represent network under cyber attacks.

3.1 System View of Live-Virtual-Constructive Platform

The LVC platform provides models for accurate cyber threat simulation at all layers of the networking stack to include passive, active, coordinated and adaptive attacks on networks with hundreds to thousands of wired and wireless components. Besides, the platform is open-ended and extensible as shown in Figure 3.1. It is configurable and may include any combination of the following:

- Live: real equipments running real applications
- Virtual: real people operating on simulated systems
- Constructive: simulated people operating on simulated systems

There are many benefits of using an LVC platform:

- It can create realistic virtual network models without having to purchase expensive equipments
- It adds realistic wired and wireless representations that can be used to assess vulnerabilities

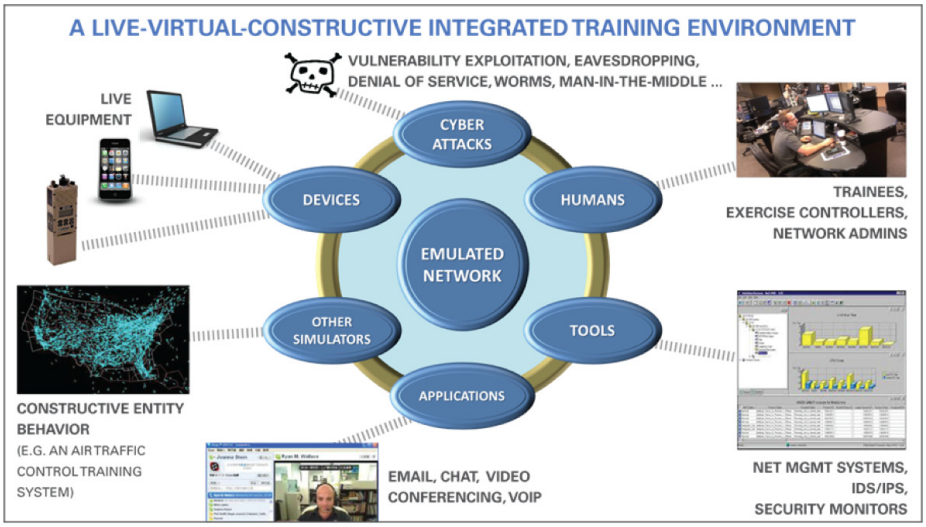


Figure 3.1: System view of Live-Virtual-Constructive platform

- It integrates specific hardware components and applications; it is adaptive to different operations environments
- It integrates with other constructive simulators
- It is easy to configure and reconfigure
- It is scalable to large network sizes and complexities
- The exercises are repeatable for training to achieve specific knowledge and skills

3.2 Usage Examples of a Live-Virtual-Constructive Platform

Figure 3.2 illustrates the usage examples of LVC emulator that combines physical machines and virtual simulated hosts. On the emulation server, we execute the simulation models of an enterprise network and a generic internet, separated by a DMZ node. The LVC framework allows real (physical) hardware to be associated with simulated nodes. In the first scenario of intrusion detection, we associate the real cyber attacker machine (at 128.4.66.1) and the real DMZ machine (at 128.4.66.10) with the Cyber Attacker and the DMZ nodes within the simulation model, respectively. Now, a human attacker could execute Metasploit

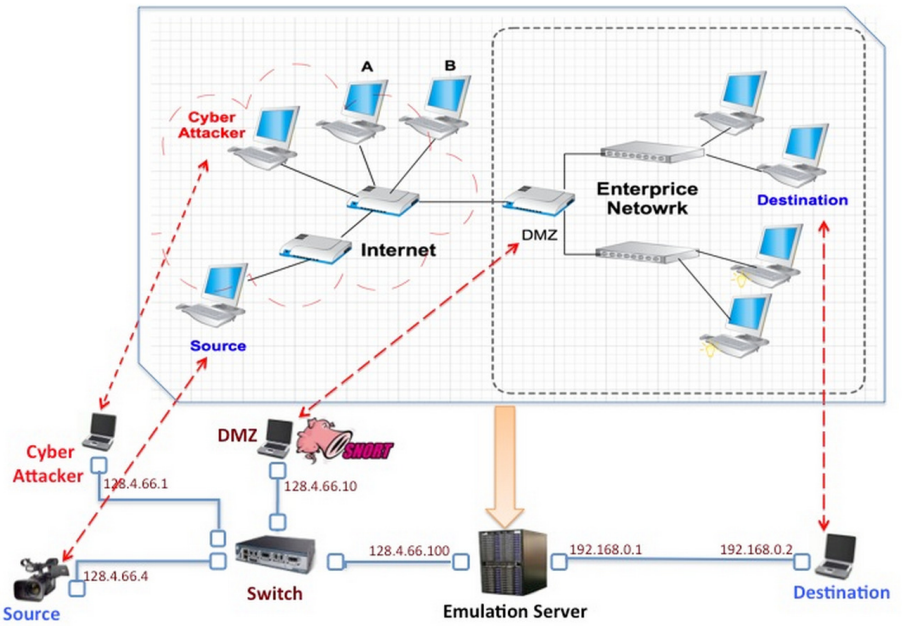


Figure 3.2: Usage example of a Live-Virtual-Constructive (LVC) platform

on the real machine to send port scans and other probes into the enterprise network. A human cyber analyst could execute SNORT on the real DMZ machine to conduct intrusion detection. Here, the real port scan and probing packets, sent from a real attacker machine, are forwarded hop-by-hop within the simulated network, and arrive at the real DMZ machine. In the second scenario of DDoS attack, we associate the real Source machine (at 128.4.66.4) and the real Destination machine (at 192.168.0.2) with the Source node and the Destination node in the simulation model, respectively. Real video traffic is generated from the Source machine, forwarded by simulated nodes, and played at the Destination machine. A botnet,formed by nodes A,B and CyberAttacker,launches a DDoS attack,and the cyber analysts at the Destination machine will observe the quality degradation of the video playback. With LVC, hundreds or thousands of nodes could be programmed to simulate large scale DDoS attacks. As cyber analysts may directly interact with real systems and/or traffic on real machines, they are immersed in realistic human-in-the-loop training by nature.

3.3 Cyber Situation Awareness Training and Assessment Framework Infrastructure

The goal of the thesis is to develop a training and assessment system for individual and team cyber situation awareness. The system infrastructure is shown in Figure3.3. As shown in the figure, lesson plan database contains different kinds of cyber attack lessons with different difficulty levels. We apply Cognitive Task Analysis on a set of tasks and use the information to generate scenarios for training purposes. For each task, we identify major events and watch list items needed for decision making. The trainees are able to tailor their watch list and triggering threshold conditions.

With the proceeding of training scenario, data such as IDS log, network flow, and trainee specified trigger alerts will be reported to the trainee. After analyzing these data, the trainee should think whether it is an attack or false alarm based on prior knowledge and decide the type of attack through attack model matching. Interactions and team discussions can be conducted through the Shared Events Viewer and team communication module. If the team members still cannot achieve agreement, the fuzzy logic based team consensus decision making module can help chose the most acceptable solution for the entire team.

The assessment metrics will include trainee response time with respect to critical cues and evaluate the actions taken or decisions made to determine potential attacks. By comparing trainees' response time and estimated attack ground truth timeline, we can identify if the response is fast or slow. The performance evaluation module can provide performance score and feedback to trainees, as well as adjust the next training lesson's difficulty level based on trainees' performance. Furthermore, Situation Awareness Global Assessment Technique (SAGAT) is used to get feedback from trainees in order to evaluate training system usability and effectiveness.

3.4 Functional View of Cyber Situation Awareness Training and Assessment Framework

Based on the LVC framework, we design a human centric training and assessment system, with high fidelity network simulation / emulation based training lessons, attack models and attack scripts, vulnerability database, performance assessment and decision support

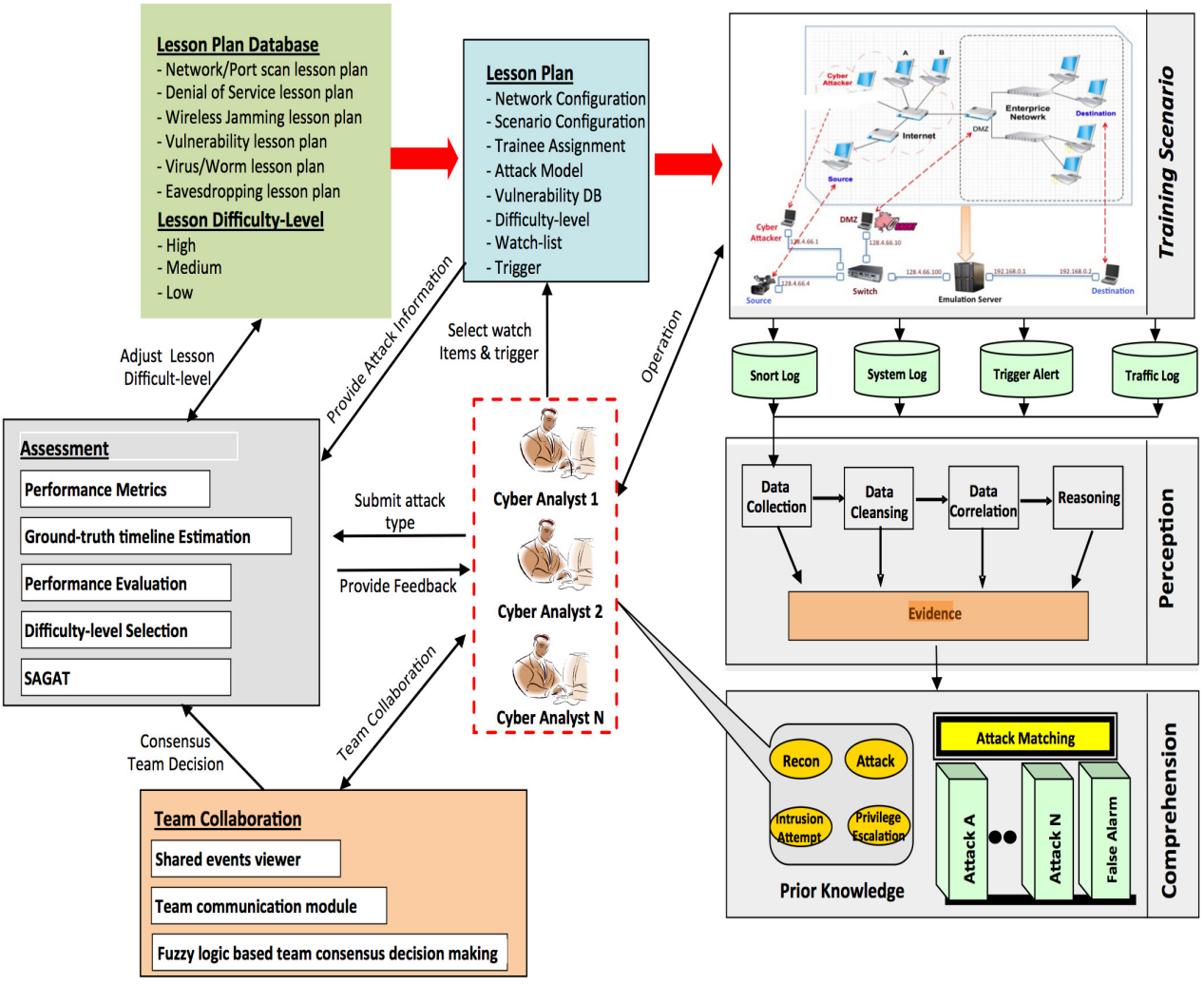


Figure 3.3: CSA training and assessment system infrastructure

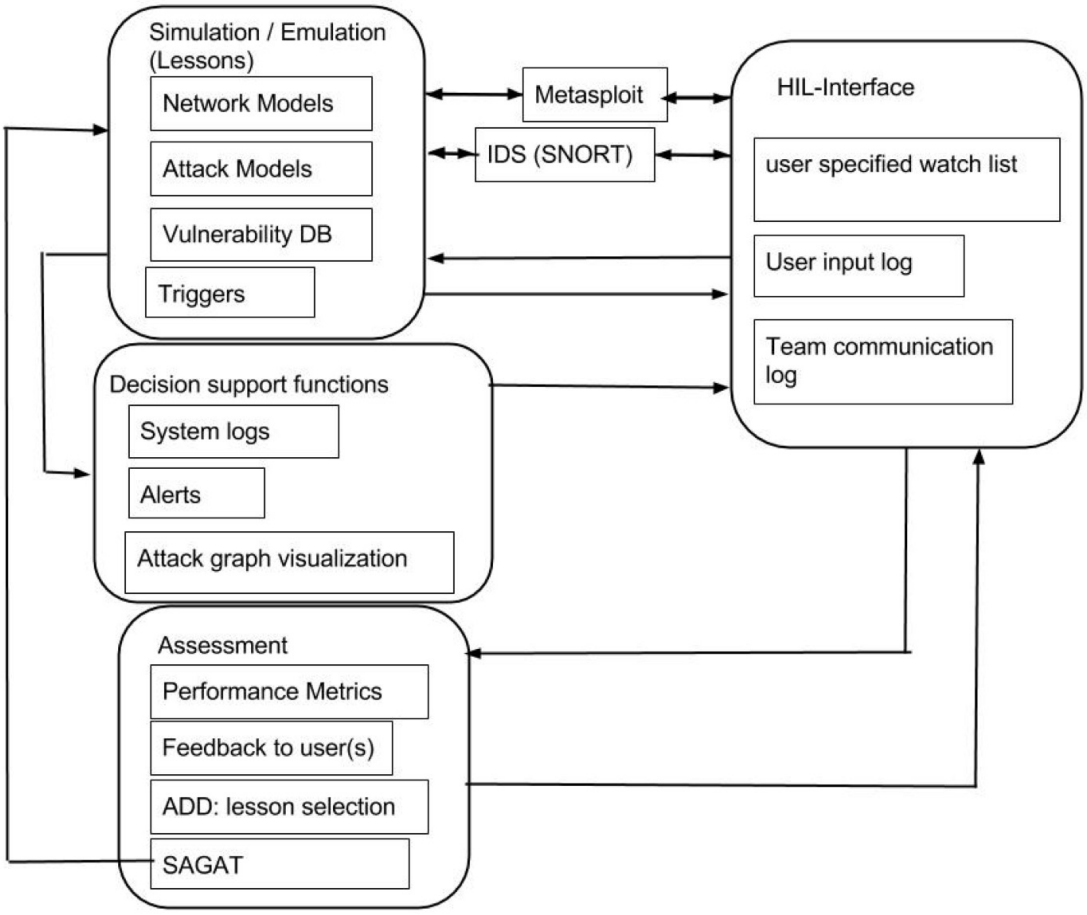


Figure 3.4: Functional view of Cyber Situation Awareness training framework

functions. Figure 3.4 shows the allocation of functions to each of the four main components: (1) Simulation/Emulation Based Training Lessons, (2) Assessment Component, (3) Decision Support Component, and (4) Human-In-the-Loop (HITL) Component:

- Simulation/emulation based training lessons: The main function of this component is provide trainees with a high-fidelity network simulation/emulation environment for conducting cyber attack and defense exercises.It contains high-fidelity network simulation models (including both protocol models and host models), a library of attack models, sample attack scenarios and attack scripts, vulnerability database, and triggers for events and logs. This network simulation/emulation approach differs from other approaches (e.g., CyberCog and idsNETS) that rely on the availability of real-world

enterprise network traffic data for training purposes. However, it does not exclude the use of historical data, if it is available. Using network simulation/emulation models provide freedom and flexibility to explore a large variety of network conditions and attack scenarios beyond pre-recorded incidents. From the human-in-the-loop interface and interaction component, the instructor and/or trainee can select the training exercises. The simulation/emulation component will then load the network configuration, vulnerability specification, time-lined traffic profile and attack sequence to simulate the behavior of the network. This component can also be connected to attack simulation tools such as Metasploit and intrusion detection systems such as SNORT. In that case, attacks launched using Metasploit will be detected by SNORT; the trainee can experiment with the use of IDS systems to detect and identify attacks. Trainees can also input the watch list he/she wants to observe and trigger conditions to the simulation component. The system logs of the watchlist and alerts will be provided to the decision making support component and made available to the trainees.

- Assessment Component: The main function of this component is to perform cognitive analysis and assessment of trainees performance, effectiveness of the tool for cognition (both individual and team), network performance, impact of cyber attacks, and effectiveness of the counter-measures. The feedback provided to instructor and trainees depend on the exercises. During the exercise, correct or failed detection of attacks will be a form of feedback to both instructor and trainees. Another assessment contains information on impact of attack (confidentiality, integrity, availability) and percentage of affected network. Team communication (data flow, number of data exchanged), time to reach consensus for decision making are also assessed. For various attacks, there might be other metrics defined for assessment. For example, for DDoS attacks, we also record the correct classification of packets so we can determine the percentage of malicious packets dropped versus the percentage of legitimate packets delivered. As training is an iterative process, the assessment component not only assesses the knowledge gained by the trainee, but also adjusts the difficulty of training

lessons accordingly based on the trainees performance.

- Decision Support Component: The main function of this component is to provide trainees with at a glance system state, to aid trainees in situation awareness in order to respond to predicted or ongoing cyber attacks. This component receives system logs, alerts from the simulation/emulation component. From the selected exercise and network configuration, it generates logical attack graph corresponding to the network configuration and takes into consideration any existing vulnerabilities on the hosts. In addition, it is synchronized with the network simulation to track progress of attacks to provide an up-to-date attack path visualization to the trainees. The detection of some cyber attacks may be automated by specifying certain rules and conditions. In that case, the decision support component will generate alert and suggest possible counter-measures. Situation awareness assessment is classified by three different levels of information. At the lowest level, situation awareness level 1 (perception), the information contains monitored raw data such as instrument readings, automated triggering of alerts etc. At the next level, situation awareness level 2 (comprehension), integration of various data elements may lead to pattern recognition and comprehension of the significance of objects and events. At the highest level, situation awareness level 3 (projection), knowing the status and dynamics of the elements (or statistical analysis) one may be able to predict future actions. The combined information visualization and decision assistance is aimed to help the trainees to reach decisions and take actions quickly.

- Human-In-the-Loop (HITL) Component: The main function of this component is to facilitate human-computer interaction. This component contains interfaces to obtain user input and to provide feedback to users as well as supporting human-computer interactions influencing the control of the exercises. HITL is a vital part of LVC platform. It is essential for training because it allows the trainees to immerse themselves in a simulated / emulated environment where their actions directly influence the outcome of events and system conditions. The trainees will interact with realistic models

and perform as they would in a real incident. The ability to train like you fight by using realistic models in a mock up is extremely effective in equipping students with knowledge, skills and experience relevant to real world proficiency at little to no risk. There is limitation on what can be automated in cyber attack detection and prevention. In most cases (of complex attacks), humans need to evaluate the information provided by the system, observe correlations, and determine the next course of action based on their judgment and experience. Intelligent systems and machine learning can only go so far in automating reasoning; humans are still needed to make critical decisions.

Chapter 4

COGNITIVE TASK ANALYSIS BASED LESSON PLANS FOR CYBER SITUATION AWARENESS TRAINING AND ASSESSMENT

4.1 Introduction

As cyber attacks are becoming more frequent and increasingly complex, the need for better training of cyber analysts to protect critical assets and ensure system security is also elevated. One issue with existing cyber security training is its relying heavily on lecture-style instructions without enough hands-on experience. Thus, it needs a realistic cyber security training system containing empirical scenarios to let cyber analysts practice deciphering sophisticated attacks.

Cyber security defense is a cognitive process for human cyber analysts. Although the cognitive work of cyber security is similar in many ways to other work involving complex systems, there are several features of cyber security that make it unique. First, the operational space is boundless with minimal geographical features. As a result, the environment from which a cyber analyst has to perceive salient cues is vastly larger and more difficult to comprehend. Second, the speed at which the cyberspace changes is much faster, where new vulnerabilities and their corresponding exploits are continuously emerging, and new offensive technologies are constantly being developed. Furthermore, modern exploits are either employed via misdirection (e.g., a DDoS attack is conducted by a botnet of compromised computers) or delivered passively via embedded malware.

Cognitive Task Analysis (CTA [59, 59]) is the extension of traditional task analysis techniques to yield information about the knowledge, thought processes, and goal structures that underlie observable task performance. The outcome of CTA describes the performance

objectives, equipment, conceptual knowledge, procedural knowledge and performance standards used by experts as they perform a task. Accurate identification of cyber security experts' cognitive processes can be adapted into training materials to teach novices how to perform like experts. In this chapter, we use the CTA approach to gain insight into the cognitive demands and workflow of cyber analysts and design cyber security training lesson plans and training workflow. Subsequently, we evaluate cyber analysts' performance based on their response time of detecting cyber attacks compared to an estimated attack ground truth timeline.

The remainder of this chapter is organized as follows: Section 4.2 describes related work. After performing Cognitive Task Analysis, we identified the steps necessary for designing cyber security training exercises in Section 4.2. Section 4.3 describes three cyber security training lesson plans. The scoring algorithm to evaluate the performance of cyber defense analysts is presented in Section 4.4. To evaluate the usability of the training system, Section 4.5 presents the questionnaire that cyber analysts are asked to answer. Finally, Section 4.6 concludes the chapter.

4.2 Cognitive Task Analysis based Approach

We design realistic lesson plans for cyber security training and assessment based on the LVC framework, which enables cyber analysts to experience cyber attacks and to learn how to detect ongoing cyber attacks. Designing cyber security lessons to involve cyber analysts in active learning requires careful planning. Cognitive Task Analysis technique [17] is a prominent approach that captures knowledge representation used by experts to perform complex tasks. We utilized a combination of three knowledge capture techniques: observing cyber security competitions, examining critical incidents, and reviewing relevant papers of structured interviews with cyber security experts and information assurance analysts [21]. We elicit the knowledge about how, when, where, and why, as we perform a cyber defense task. This knowledge can be applied into design consideration for cyber security training lesson plans.

Notice that human cyber analysts have to check thousands of events each day from many sources such as system logs, configurations, traffic logs, IDS log, and audit logs in order to determine whether there are real attacks or false positives present; therefore, they could likely be soon overwhelmed by tremendous data and forced to ignore potentially significant evidence introducing errors in the detection process. In order to solve the tremendous cognitive demand faced by cyber analysts, we identify and design watch list items related to cyber attacks. Cyber analysts can tailor their own watch list items and triggering thresholds in order to detect cyber attacks faster.

Six steps necessary for building training lessons is as follows:

1. Previous related work review
2. Training objective definition
3. Training scenario creation
4. Cyber analyst watch-list definition
5. Cyber analyst response recording
6. Performance assessment

Based on the design steps, the training workflow is shown in Figure 4.1, which contains the following steps:

Step 1: Instructor creates a lesson plan for the cyber security training that includes a cheat sheet for the cyber attack/defense aspect based on the lesson objective. The Cheat Sheet includes the watch list items critical to the cyber attack and the attack ideal timeline denoting the attack start and success time. Cyber analyst should react to the cyber events in simulation and perform certain actions that demonstrate his/her understanding of cyber attacks.

Step 2: Instructor sets up training scenario with the tool providing the widgets to enable the instructor to enter in the information from the cheat sheet.

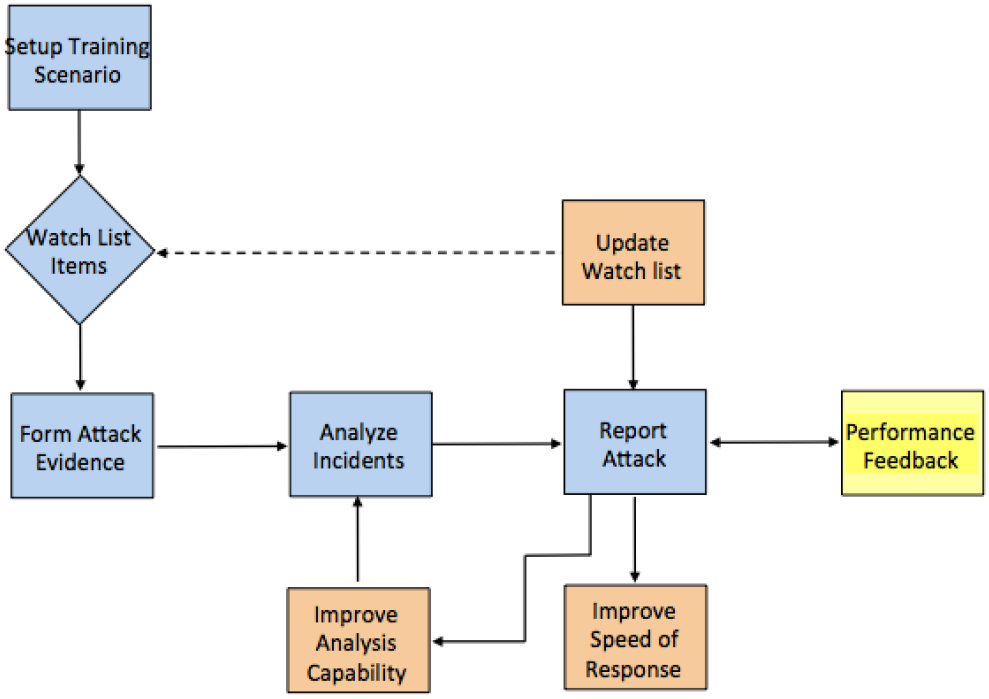


Figure 4.1: Workflow for CSA training system

Step 3: When training scenario begins, the specified trigger alert and other log data specified by cyber analyst will be sent to cyber analyst side. After analyzing these data, cyber analyst should think whether it is an attack or false alarm based on prior knowledge and decide the type of attack through attack model matching.

Step 4: During the training, with cyber analyst's actions being logged continuously, the training system can determine whether the response actions of cyber analyst are following the ideal timeline enumerated by instructor in the cheat sheet.

Step 5: Based on cyber analyst's response and the ideal timeline, the score for cyber analyst will be computed using devised scoring mechanisms, and provided to cyber analyst as part of after action review.

Step 6: After obtaining performance assessment report, cyber analysts should think about selecting different watch list items or improving analysis capability for the next lesson.

Based on the tailored lesson plans, cyber analyst will learn the knowledge required to monitor network conditions and identify ongoing attacks. After completing the cyber

security training, cyber analysts will be able to do the following with respect to a given set of known attacks:

- List the relevant parameters to monitor and know the characteristics of these parameters under normal and abnormal operations.
- Recognize symptoms of network attacks. Specifically, cyber analyst will be able to isolate common characteristics of network under attack and be able to distinguish the characteristics that are particular to each attack.
- Given a particular set of current conditions (monitored parameters), be able to analyze what kind of attack is occurring and how the attack was launched.
- Demonstrate proper procedure of remedial actions, including selection of countermeasures to apply and where in the network to apply them.

4.3 Cyber Security Training Lesson Plans

Guided by the lesson design steps and goals of cyber security training, we describe three cyber security lesson plans [40]: port/network scanning, denial of service, and wireless jamming.

4.3.1 Port Scan Attack Lesson Plan

Usually an attacker may attempt to obtain information concerning a network in order to choose its course of malicious actions against the network. The lesson goal is to familiarize cyber analysts on how to detect network and port scan attack. Specifically, after network scan, an attacker is able to discover the number of hosts, the IP addresses and the network topology. The next step an attacker may take is to perform a port scan [73] to discover which hosts are critical and what services are running on the various hosts. The obtained information can be used by an attacker to plan attack attempts targeting various vulnerabilities.

An example of lesson scenario on how to perform network and port scan is illustrated in Figure 4.2. Initially, the core gateway router is configured without firewall. Before

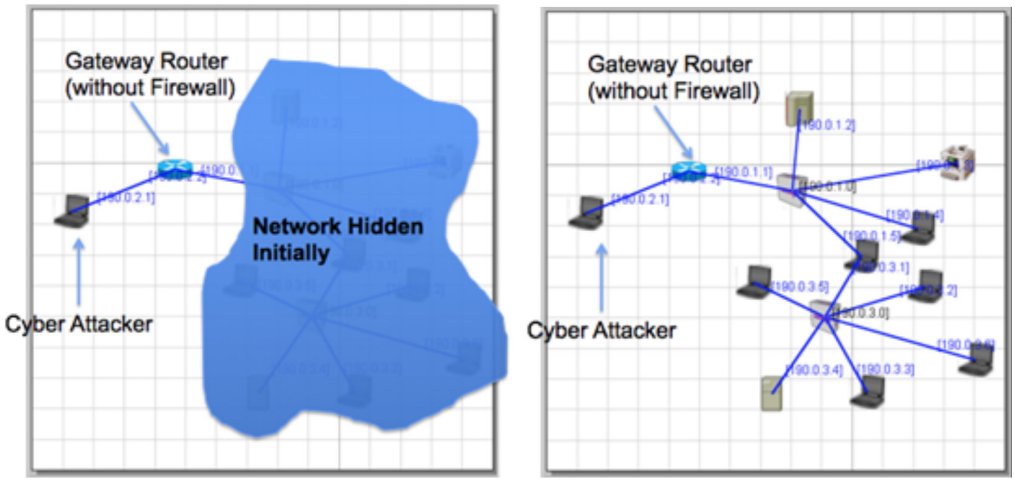


Figure 4.2: Port/Network Scan Attack lesson scenario

performing a network scan, the "inside" network is not visible to the outside world. The attacker then attempts to obtain information on the internal network by launching a network scan. Without firewall protection at the router, the network scan is able to discover the number of hosts, IP addresses and the network topology by sending a bunch of probe packets. Once an attacker learns the IP addresses of hosts and network connectivity, he/she can launch port scan to discover applications such as web server, FTP server running on the hosts and other devices connected to the hosts such as printer.

For the purposes of facilitating cyber analysts' analysis, we define network scan as a procedure for identifying active hosts on a network [1]. After observing the network traffic that contains certain numbers of distinct probes within several seconds from a single anomalous source, it might be a potential network scan attack. The port scan is defined as an attack that sends client requests to a range of server port addresses [8] on a host with the goal of finding open ports and corresponding services running on the ports. By observing several requests to a range of port addresses, it might be a potential port scan attack.

4.3.2 Denial Of Service Attack Lesson Plan

Denial of service (DoS [86]) attacks are characterized for attacking with the purpose of preventing victim computing systems or networks from providing services to legitimate

users. Such attacks aim at stopping the system from working properly through the recruitment of a large number of "zombie" machines to send high volumes of ordinary traffic to a target machine or network. The DoS lesson objective is to train cyber analysts to understand DoS attacks and their detection methods. The issue of DoS protection and mitigation will be discussed in the lesson on Firewall. In this lesson, we will study three types of DoS attacks:

- Basic DoS attack: the attackers send large volume of UDP traffic to the victim host or network. Such traffic consumes network bandwidth, buffer memory as well as CPU resources.

- TCP SYN DoS attack: the attackers send TCP SYN packets to the victim host. Each TCP SYN packet opens a new TCP connection at the victim computer, thus consuming transport-layer buffer memory.

- IP Fragmentation DoS attack: the attackers send partially fragmented IP packets to the victim host. The victim computer buffers these fragmented packets and wait for remaining segments, thus consuming network-layer buffer memory.

To identify DoS attacks, cyber analysts need to first differentiate attacks from normal bursts of network requests in order to reduce rate of false positive.Therefore, we need to compare it with behaviors of normal requests burst, analyzing information via system/network logs for determination. It is important to know what kind of information are being request for and what type of requests they are. Generally, requests from DoS attacks share the same target. For example, a great many of requests, possibly tens of thousands or more, are trying to access a specific URL or port. Although sometimes under DDoS attacks there could be multiple targets. It is still possible to find that portions of the requests are aiming for same target.

Once cyber analysts identify the incoming requests are indeed results of DoS attacks, the next information cyber analysts need to acquire is what kind of DoS attack it is and what attack techniques it is employing. Since cyber analysts already know about the details of the requests from the system logs, cyber analysts can generally determine the type of DoS attacks

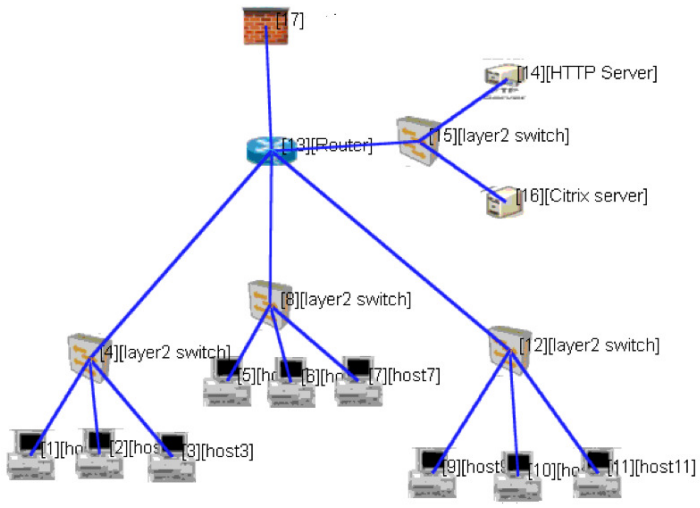


Figure 4.3: DoS Attack lesson scenario

and then take corresponding defending actions. As introduced previously, cyber analysts are considering three DoS attack scenarios, utilizing basic UDP traffic, TCP SYN requests, and IP fragmentation separately.

Basic DoS attacks involve sending a large volume of traffic to the host, exhausting the host's processing and memory resources and making unable to serve more normal traffic. As a result, there is a sharp increase in the memory and CPU usage of server hosts.

TCP SYN attack happens when attackers send a flood of TCP/SYN packets with faked sender addresses. Each packet is treated as a connection request and the server maintains a half-open connection for each request. The server send TCP/SYN ACK packets back to the faked senders and waits for the responses. Since the sender addresses are faked, the responses will never come. And the number of half-open connections quickly saturate the buffer resources of the server, rendering it unable to serve future legitimate requests.

Another technique of DoS attack is via IP fragmentation. As we know, the packet size must not be larger than the network's Maximum Transmission Unit (MTU). If the data to transmit is larger than the size of MTU, then it has to be fragmented into multiple pieces of small packets, each of which contains a portion of the data. And these packets will be reassembled at the receiver. The attackers, however with the help of softwares, could force the system to send packets size larger than the network's MTU and crash victim hosts.

Figure 4.3 shows an example DoS exercise scenario, which includes nine "zombie" machines that can send high volumes of ordinary traffic to the target victim machine. An attack targeting at the HTTP server typically involves sending a large number of requests, each of which consumes significant resources. This then limits the ability of the server to respond to requests from other users. Besides, Web requests may make database queries. If costly query can be constructed, then a large number of requests could trigger these queries that would severely overload the server and limit its ability to respond to legitimate requests. The attackers starts either one of the three types of DoS attacks. The magnitude and difficulty of the exercise scenario can be controlled by the number of "zombie" machines as well as traffic volume.

To perform DoS attack detection [26, 51], traffic flow monitoring is the key. Such traffic flows are characterized by a set of watch list. Example watch list items include CPU and memory resource usage, number of incoming flows, aggregated traffic rate as shown in Table 4.1.

Table 4.1: DoS exercise watching list

CPU Utilization	CPU utilization is calculated by CPU load as a percentage of the CPU capacity
Memory Usage	Memory consumed during certain period
Network Traffic	- Network-FIFO,Total packets queued - Network-FIFO,Total packets dropped - Network-FIFO,Average queue length - Network-FIFO,Peak queue size - Network-FIFO,Longest time in queue - IP,Number of IP Fragments received - IP,Number of IP Fragments dropped - UDP, Number of packets received - UDP, Number of bytes received - TCP, Number of packets received - TCP, Number of bytes received

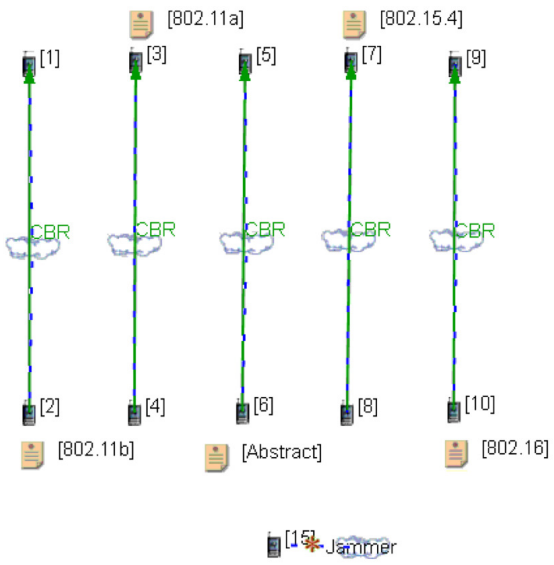


Figure 4.4: Wireless Jamming Attack lesson scenario

4.3.3 Wireless Jamming Attack Lesson Plan

Wireless jamming attacks [108] can be accomplished by an adversary emitting radio frequency signals that do not follow an underlying MAC protocol. Wireless Jamming attacks can be viewed as a special case of DoS attacks, which prevent or inhibit the normal use through flooding a wireless network with useless information. The lesson objective is to train cyber analysts to understand the jamming attacks and their detection.

Figure 4.4 depicts an example wireless jamming lesson scenario, which includes five pairs of senders and receivers with different MAC protocols. The jammer is host 15 who can transmit random radio signals to increase the background noise in the channel. Thus, when packets reached at the receiver side, they cannot be correctly decoded and hence are dropped. Another jamming type is to constantly emit random semi-valid packets to keep the medium busy all the time, preventing the honest nodes from switching from the listening mode to the transmitting mode. The magnitude of the exercise scenario can be controlled by the number of jammers and their jamming frequency.

The complexity of detecting jamming attack is the fact that there might be multiple reasons which cause decreased performance of wireless network except intentional attacks driven by adversaries. Thus, it is critical to distinguish between poor network conditions due

to natural environmental influences and intentional jamming attacks. No single measurement is capable of detecting all kinds of jamming attacks [93, 32], thus, we teach cyber analysts to combine several metrics to detect jamming attacks as following:

- Signal Strength: If jamming attack exists in communication channel, the signal-to-noise ratio (SNR) decreases and frame decoding at receiver node becomes error prone. By gathering enough noise level measurements during a time period prior to jamming, cyber analysts can identify normal SNR levels in the network.

- Carrier Sensing Time: The jamming attacks can keep channel busy that results in longer channel waiting time at the sending side. Carrier sensing time denotes the amount of time it spends waiting for the channel to become idle. By keeping track of carrier sensing time during normal traffic operations, cyber analysts can determine normal carrier sensing time range.

- Packet Send Ratio: A jammer can interfere with legitimate wireless communications by preventing a real traffic source from sending out packets. The packet send ratio can be defined as the ratio of packets that are successfully sent out by a legitimate traffic source compared to the number of packets it intends to send out at the MAC layer.

- Packets Delivery Ratio: The packet deliver ratio can be calculated as the ratio of packets that are successfully delivered to a destination compared to the number of packets that have been sent out by the sender. Another measure method is calculating the ratio of the number of packets that pass the Cyclic Redundancy Check (CRC) check with respect to the number of packets received.

- Energy Consumption Amount: The victims may receive large volume flooding packets with useless information from jamming attackers. It consumes valuable energy resources in sensing the channel and processing the traffic in the channel. The energy consumption amount can be defined as the approximate amount of energy consumed in a period of time.

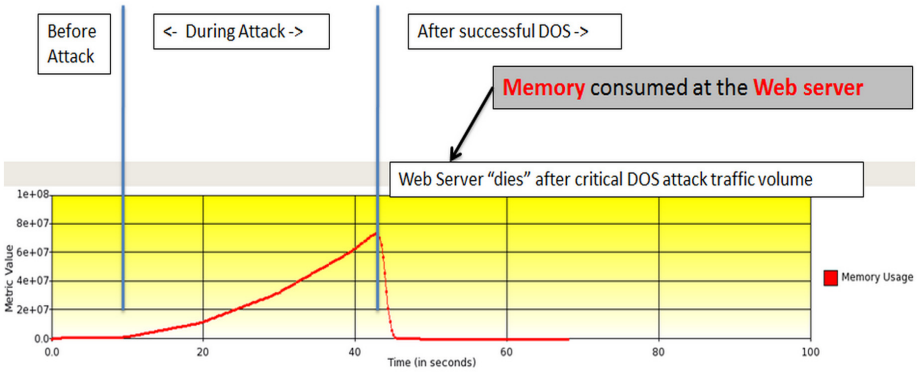


Figure 4.5: Use memory usage metric to detect DoS

- Mobility: For wireless communication, the signal strength is fading with a power of two with respect to the distance between sender and receiver. Thus, node's mobility can also cause performance deterioration beside jamming attacks.

4.4 Performance Metrics and Scoring Algorithms

To monitor the activities of and provide feedback to a cyber analyst during training sessions, we adapt the method of timeline analysis. The ideal timeline of a training exercise is gauged based on the specific attack scenario and its configuration. After a training exercise starts, all of a cyber analyst's actions are continuously logged so that the training system can determine whether actions taken by the analyst follow the ideal timeline and match the expected activities. This evaluation can be provided as feedback to the cyber analyst during training. For instance, if an analyst fails to identify attacks in time, the system can proactively provide hints to the analyst, or share the views of other cyber analysts. Trainees may also ask for a hint from the instructor. The training performance of a cyber analyst is evaluated based on his/her response time of correctly identifying specific cyber attacks.

Figure 4.5 depicts how the measurement of memory usage can be used to characterize the ideal timeline for a DoS attack. Two memory usage thresholds divide the time period into three phases: before attack, during attack, and after a successful DoS attack. Based on the pre-defined memory usage thresholds and the DoS training lesson's configuration (such as the number of packets to be sent, the frequency of sending packets, the start and end times),

a DoS attack's start time and its time of successful attack can be determined. Similarly, the method can be applied to other DoS metrics such as CPU usage, number of incoming flows, and aggregated traffic rate to generate their corresponding timelines. By combining the timelines together, an ideal timeline for the DoS attack can be generated.

Based on the response of cyber analysts and the ideal timeline, scores for the performance of cyber analysts can be computed using the devised scoring mechanisms, and they can be provided to the cyber analysts as one component of their after action review. By comparing cyber analysts' response times against the ideal timeline, we can determine whether a cyber analyst responds in a timely manner. For instance in Figure 4.5, assuming a DoS attack starts at a time of 10 seconds and sustains for 35 seconds, and the victim host shuts down at a time of 45 seconds, a cyber analyst has a time window of 35 seconds to identify the ongoing DoS attack. If a cyber analyst identifies this DoS attack at the 20 second mark, the cyber analyst's response is considered fast enough to score high on the exercise. In contrast, if a cyber analyst does not identify the DoS attack until the victim host shuts down, no points will be given.

To evaluate the performance of cyber analysts, a set of performance metrics has been adopted:

- Lesson magnitude and difficulty levels (W_D): stands for the severity of attacking or the difficulty of achieving an attack goal. A scenario's difficulty level is specified in one of three categories: High, Medium, or Low.
- Response time (W_T): measures cyber analysts' responsiveness to correctly recognizing cyber attacks.
- Correct detection of attacks (W_C): identifies the existence of a real attack and its type.
- Damage impact (W_I): measures attacks' impact on victim's confidentiality, integrity, and/or availability.

Based on the performance metrics, the score of cyber analysts can be calculated by the following formula:

$$Score = W_D * (\sum_{k \in \{T,C,I\}} W_k * K_k)$$

where $K_k, k \in \{T, C, I\}$ is the weight factor for each performance metric. Notice that the lesson difficulty level W_D is separated from other performance metric during score calculation. This is because the more difficult of the lesson, the higher score is given since trainees have to spent more time and effort to performance the defense task. For the purpose of consistent computation, each weight factor is normalized to the value between 0 and 1. Take difficulty level W_D as an example, training lesson labeled with "Low" difficulty has weight factor value 0.4 and "Medium" difficulty training lesson is given weight value 0.7.

4.5 Evaluate Cognitive Validity of Training

In order to evaluate the usability of the training system and its effectiveness, Situation Awareness Global Assessment Technique (SAGAT) is used. SAGAT covers the three levels of CSA including Level 1 (perception of data), Level 2 (comprehension of meaning), and Level 3(projection of the near future). Typically, a set of CSA queries regarding the current situation is asked and participants are required to answer each query therein based upon their knowledge and understanding of the situation at that point. The detail of SAGAT questionnaire is in Appendix A and the example questions to be asked are as follows:

1. CSA related queries

    (a) An IDS alert based on traffic from 192.168.2.42 destined to 192.168.1.252 is best classified as?

    (b) Which watch list item is abnormal?

    (c) Is it an attack or false alarm?

    (d) What is the impact for the current attack? Any confidentiality, integrity, or availability loss?

    (e) What actions should be performed to stop this attack?

2. Participant satisfaction

    (a) Is the training tool easy to use?

    (b) Is the information displayed in a way that is easy to comprehend?

    (c) Does the tool provide information needed to achieve lesson goals?

    (d) Are the lesson contents at the appropriate difficulty level for the cyber analysts?

    (e) Are the hints useful?

3. Knowledge acquisition

    (a) Does the cyber analysts grasp the main objectives of the lesson?

    (b) Does the lesson learned lead to intended decision-making skills?

4. Behavior changes

    (a) How does the acquired knowledge affect the cyber analysts in future operations?

    (b) Will the cyber analysts be able to detect and identify DoS attacks faster?

4.6 Conclusion

Accurate identification of experts' cognitive processes can be adapted into training materials. In this chapter, we described three cyber security training lesson plans: port/network scanning, denial of service, and wireless jamming after performing cognitive task analysis. We also defined the metrics for performance evaluation and the corresponding scoring algorithms.

Chapter 5

DIFFICULTY-LEVEL METRIC FOR CYBER SITUATION AWARENESS TRAINING AND ASSESSMENT SYSTEM

5.1 Introduction

In order to conduct effective cyber security training, it is critical to differentiate attack scenarios of various difficulty levels in order to accommodate defense analysts' skill levels. While standard techniques exist for measuring the relative difficulty to exploit an individual's vulnerability, it is challenging to answer the fundamental question "whether one scenario containing several vulnerabilities is more difficult than another one."

Simple ways of aggregating individual vulnerabilities, such as taking the average or maximum values, may foster misleading results. An attack graph [57] can be used to represent a collection of possible penetration steps in attack scenarios. Each penetration step is a sequence of actions taken by an intruder aiming to achieve a particular goal. An attack graph for a network shows all the hosts that can be compromised by an attacker starting at a specific location and the sequences of actions that can exploit the vulnerabilities in the network. Therefore, attach graphs can reflect the casual relationship between vulnerabilities.

In this chapter, we describe a probabilistic metric for measuring the aggregation effect of individual vulnerabilities and producing quantitative difficulty levels for attack scenarios[41]. More specifically, we adapt the existing work of attack graphs by applying Bayesian Reasoning to combine the individual vulnerabilities into a probabilistic value that represents the attack success likelihood. This combination is based on the causal relationships between vulnerabilities in an attack graph. One major limitation of using Bayesian Reasoning is that it does not allow cycles, which are common in attack graphs. We identify different types of cycles in attack graphs and design an efficient algorithm to remove

them while keeping cyclic influence in the probability calculation. Finally, we use empirical scenarios to evaluate our metric calculation method.

This chapter proceeds in Section 5.2 with a review of the related work. Section 5.3 introduces a motivating example and examines how to handle cycles in attack graphs. Then, the cycles removing algorithms and the proposed difficulty level metric are defined. Section 5.4 presents the empirical evaluation. Finally, Section 5.5 summarizes this chapter.

5.2 Background and Related Work

Attack graphs [87, 2] have been used to represent all possible paths an attacker can take to achieve certain goals by exploiting system vulnerabilities. Much work has already been done in analyzing network configurations and identifying network vulnerabilities to construct attack graphs [43, 46].

Early efforts on building attack graphs are presented by Network Security Planning Architecture (NetSPA [3]). The authors use attack graphs to model adversaries and the effect of simple counter measures. It creates a network model using firewall rules and network vulnerability scans. It then uses the model to compute network reachability and attack graphs representing potential attack paths for adversaries exploiting known vulnerabilities. This discovers all hosts that can be compromised by an attacker starting from one or more locations. NetSPA typically scales as *O(nlogn)* as the number of hosts increases.

In [88], an attack graph is generated based on a model checking method. In this type of attack graph, each vertex represents the network state and each edge represents a transition from one network state to another state when attacking actions are applied. The central idea is to use an exploit dependency graph to represent the pre and post conditions for an exploit. Then a graph search algorithm is used to chain the individual vulnerabilities and find attack paths that involve multiple vulnerabilities. The drawback is that it is not very scalable with respect to the number of nodes in the network.

A logic programming-based attack graph called MulVAL is introduced [72]. The key idea is that most configuration information can be represented as Datalog tuples, and most attack techniques and OS security semantics can be specified using Datalog rules. After

performing analysis, the trace of the evaluation is recorded and sent to a graph builder, where the logical attack graph is generated automatically.

General reviews of security metrics are given in [48, 98]. Phillips et al. [75] utilize graph path metrics such as the shortest path, the number of paths, and the mean of path lengths to quantifiably rate network security. Idika and Bhargava [42] observe the shortcomings of these path metrics and describe a complimentary path metrics suite, such as normalized mean of path lengths, standard deviation of path lengths, and median of path lengths. Noel and Jajodia [68] present a suite of metrics for measuring network-wide cyber security risk including the size of the attack graph, the topology graph theoretic properties such as connectivity, cycles, and depth of the attack graph.

The Common Vulnerability Scoring System (CVSS [83]) has been designed to provide a standard way to rate individual vulnerabilities. The vulnerabilities are assigned numerical scores based on their exploitability and potential impact on confidentiality, integrity, and availability. The key limitation of such individual vulnerability scores is that it is not possible to capture the cumulative effect of these vulnerabilities in network.

Since attack graphs represent the logical relationship among possible attack steps, it is natural to integrate attack graphs with CVSS scores to calculate the network security metrics. Frigault et al. [28] model attack graphs as special Bayesian Networks (BN). Based on the individual probabilities and Conditional Probability Tables (CPT) for each node, it can calculate the attack success likelihood. The drawback is that cycles exist in attack graphs and the BN method cannot be applied directly to it. Wang et al. [101] define the basic metric and give a meaningful interpretation to the metric. Then, the authors identify the presence of cycles in the attack graphs and extend the definition accordingly to propagate probability over cycles. In [38, 110], the authors utilize existing MulVAL attack graphs and apply probabilistic reasoning to produce an aggregation metric. It is similar to our work except that they only use CVSS's Access Complexity as node's metric and map them to three fixed probability such as low to 0.9, medium to 0.6, and high to 0.2. Besides, their approach utilizes the concept of d-separation in BN inference instead of BN reasoning.

Besides the attack graph-based metrics, Manadhata has brought a new concept called

Attack Surface [60, 61]. A general analysis of a system can determine the system's attack surface based on the tuple $\langle methods, channel, data \rangle$. The larger the attack surface, the less secure the system. However, this kind of analysis is often not feasible due to required source codes and expert knowledge of the system. However, in the real world, it is hard to get the source codes and resource constraints may preclude such kind of expert knowledge.

5.3 Difficulty-Level Metric

Due to the scalability advantage of a MulVAL attack graph [71], we utilize it as the structural basis for aggregating individual vulnerability into an attacker's success likelihood of achieving attack goal. Our approach could be easily adapted to other tools that produce attack graphs with similar semantics.

5.3.1 Motivating Example

Figure 5.1 depicts an example enterprise network [37], which includes a Demilitarized Zone (DMZ) and internal servers. The Web Server and VPN Server are directly accessible from the Internet through the firewall. The Web Server can access the File Server through the NFS file-sharing protocol. The Database Server contains enterprise information, which can only be reached through the Web Server and internal servers.

The Web Server contains vulnerability CVE-2006-3747 in the Apache HTTP Server. Depending on the way in which Apache was compiled, this vulnerability can be triggered by remote attackers to potentially allow execution of arbitrary code. Internet Explorer (IE) has vulnerability CVE-2009-1918, by which user workstations could be compromised if users access malicious online content through IE. MySQL in the Database Server has vulnerability CVE-2009-2446, where attackers can exploit this vulnerability by formulating specially crafted SQL commands to gain administrator privilege to execute arbitrary code.

Based on this network configuration and host vulnerabilities, the corresponding logical attack graph generated by MulVAL is presented in Figure 5.2. The MulVAL attack graph consists of two types of nodes: the privilege nodes and the attack step nodes. The privilege nodes are represented in the graph as square shaped OR-nodes. Each node describes a single

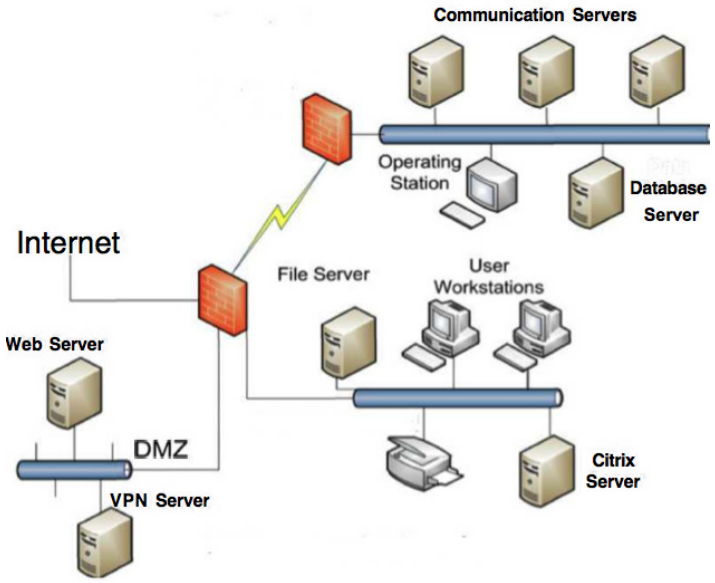


Figure 5.1: Example enterprise network

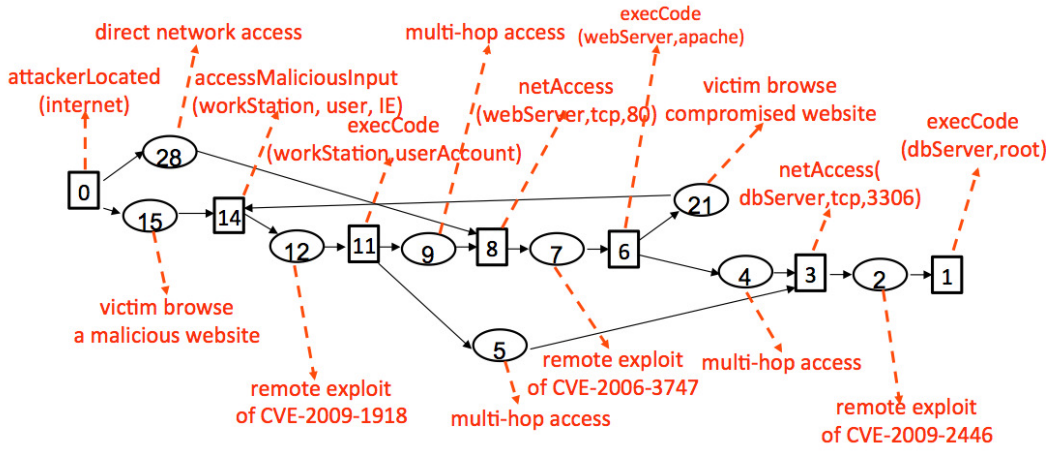


Figure 5.2: Example attack graph

network privilege, which is acquired based on the current network configuration or achieved by any one of its predecessors attack steps. The attack step nodes, represented in the graph as oval shaped AND-nodes, show the attack step that can be performed only when all of its predecessors are satisfied. Notice that an attack step AND-node is only preceded by privilege OR-nodes and always has exactly one successor OR-node. An OR-node, however, may have multiple successor AND-nodes, representing different attack steps requiring this privilege as a preceding condition. Labels of the nodes have a generic format of *predicate(parameters)*. Take attacking goal as an example, node 1 is labeled "*execCode(dbServer,root)*", where "*execCode*" is a predicate and "*dbServer*" and "*root*" are two parameters, meaning that an attacker can execute arbitrary code with root privilege on host dbServer.

As shown in the attack graph, there exist alternative sequences of attacks in order to achieve the goal of gaining privileges to execute arbitrary code on the Database Server. For example, as denoted in Node 12 "*remote exploit of CVE-2009-1918*", an attacker can compromise workstation by exploiting this vulnerability when the users access malicious online data through Internet Explorer. Using it as a stepping stone, the attacker can launch an attack to exploit Web Server's vulnerability. Finally, the attacker can attack the database server to gain administrator privilege from there.

5.3.2 Cycles in Attack Graph

One major complication of applying Bayesian Reasoning [6] on Attack Graphs to define cumulative metric is that Bayesian Reasoning does not allow directed cycles. However, cycles naturally exist in attack graphs. Take the attack graph in Figure 5.2 as an example, it contains one cycle 6-21-14-12-11-9-8-7-6.

To illustrate the different types of cycles in MulVAL attack graphs, we use a small hypothetical attack graph as shown in Figure 5.3. The attack graph on the left shows the type of cycle that can be completely removed. Cycle A_1-P_2-A_2-P_3-A_1 is formed through semantic reasoning, but, none of the states on the cycle can ever be reached by attackers. Specifically, attack step A_1 can only be carried out when both predecessors P_1 and P_3 are satisfied. However, privilege P_3 can only be achieved through attack step A_2, which relied

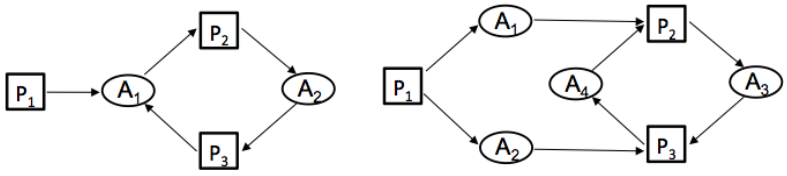


Figure 5.3: Cycles in attack graphs

on P_2 as the precondition that can only be obtained through attack step A_1. In other words, attack step A_1 and privilege P_3 relies on each other as preconditions. Therefore, neither A_1 nor P_3 can ever be successfully executed. This type of cycle can be completed removed from the attack graph when calculating the cumulative scores.

The attack graph on the right of Figure 5.3 shows that some cycles cannot be removed due to the fact that some states on the cycle can indeed be reached. Take cycle P_2-A_3-P_3-A_4-P_2 as an example, every node can be reached. Since privilege P_2 is an OR-node, it can be achieved by either attack step A_1 or A_4. Similarly, privilege P_3 can also be obtained by two different attach paths P_1-A_1-P_2-A_3-P_3 and P_1-A_2-P_3. This kind of cycle cannot be directly removed, as it can influence the result of probability calculation.

5.3.3 Handling Cycles

In order to utilize Bayesian Reasoning [6] to calculate attack success likelihood, we design an algorithm to convert a general attack graph to an directed acyclic graph. To do so, we first formalize the MulVAL attack graph with the following definition:

Definition 1. *An attack graph is a directed graph* $G = (G_E \cup G_P, E_A \cup E_O)$*, where* G_E *is the set of attack step nodes, and* G_P *is the set of privilege nodes.* $E_A \subseteq G_P \times G_E$ *is the set of edges denoting the AND relation and* $E_O \subseteq G_E \times G_P$ *is the set of edges denoting the OR relation.*

Typically an attack graph contains several privilege nodes as an attacker's starting points. For the purpose of computational efficiency, we add a virtual privilege node G_R as

the parent of all the attack starting points and starts computation only from this super source. Cycles in the graph are identified using Tarjan's algorithm [35] for strongly connected components. Given the cycle path sets, if a path starts from an attack step node and ends at the same attack step node, we can judge that this cycle can be removed completely. Otherwise, the cycle cannot be directly removed.

We describe a straightforward approach to unfolding a directed cyclic graph into an equivalent tree. As described in Algorithm 1, it starts from extracting a covering tree from the super source node G_R. Then each edge that is not included in the covering tree is added as a clone edge to generate the equivalent tree. Take attack graph in right side of Figure 5.3 as an example, the corresponding covering tree and equivalent tree of attack graph are shown in Figure 5.4.

Algorithm 1: REMOVE CYCLES IN ATTACK GRAPH

Input: The directed cycle attack graph $G = (G_R, G_E \cup G_P, E_A \cup E_O)$
Output: The corresponding attack graph G' without cycles

$n \leftarrow G_R$ {Begin with super source node}
Extract a covering-tree $T_C = (G_R, G_E \cup G_P, E_C)$ having G_R as the start
$V_0 \leftarrow G_R \cup G_E \cup G_P$
$E_0 \leftarrow E_C$
$E_R \leftarrow E_A \cup E_O - E_C$
while $E_R \neq \{\}$ **do**
    $\langle v_1, v_2 \rangle \leftarrow E_R.pop()$
    $v_{new} \leftarrow$ a new clone for v_2
    $V_0 \leftarrow V_0 \cup v_{new}$
    $E_0 \leftarrow E_0 \cup \langle v_1, v_{new} \rangle$
return $T_0 \leftarrow (V_0, E_0)$

5.3.4 Calculating Probability of Achieving Attack Goal

In order to calculate the likelihood of an attacker achieving certain attack goal, first, the probability of individual vulnerability associated with each attack step node is calculated

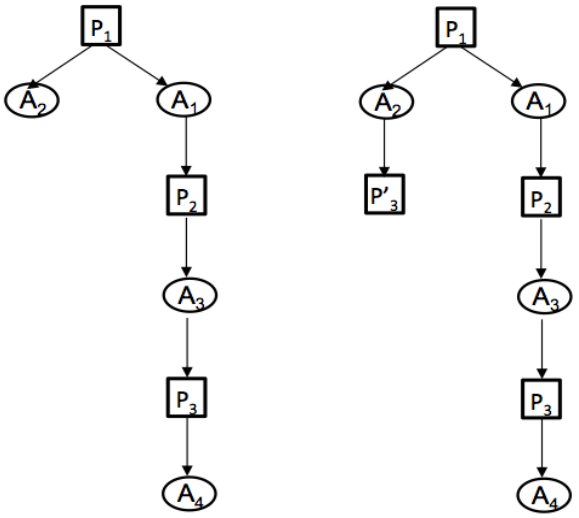


Figure 5.4: Covering tree and equivalent tree of the attack graph

according to the CVSS exploitability score. Second, applying Bayesian Reasoning on acylic attack graph to compute the probability of likelihood to achieve the attack goal.

CVSS provides an open framework for rating vulnerabilities. The CVSS basic score is calculated based on exploitability metrics such as Access Vector, Access Complexity, and Authentication.We take the exploitability score in CVSS and map it to a conditional probability of exploit success. For example, vulnerability CVE-2009-2446 in node 2 of Figure5.2 has a CVSS score of 8.5. We convert this score into a probability by dividing it by 10. Each vulnerability's preconditions, which are privilege nodes without predecessor in an attack graph, are assigned probability 1.

Since an attack graph has been remodeled as a directed acyclic graph and each node is assigned with probability value, the Bayesian Reasoning method [6] can be directly applied to calculate the probability of attaining the final attack goal. The Bayesian Networks full join distribution is calculated as the product of the local conditional distributions:

$$P(X_1, X_2, ..., X_n) = P(X_1|X_2, ..., X_n)P(X_2|X_3, ..., X_n)...P(X_n)$$

After obtaining the probability of achieving an attack goal, we rank a scenario' difficulty level into three categories: High, Medium, and Low. Scenarios are labeled "High"

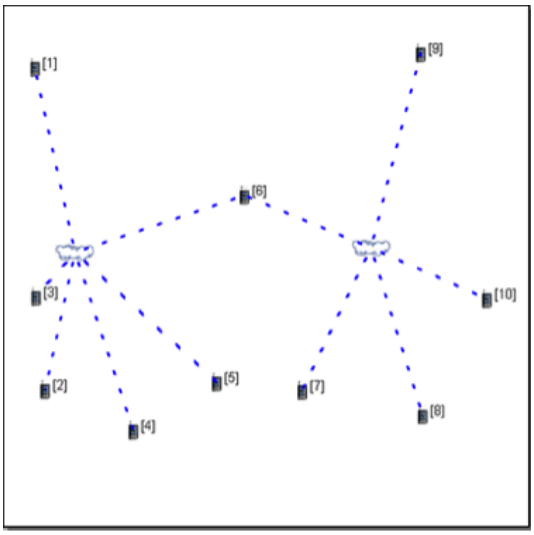


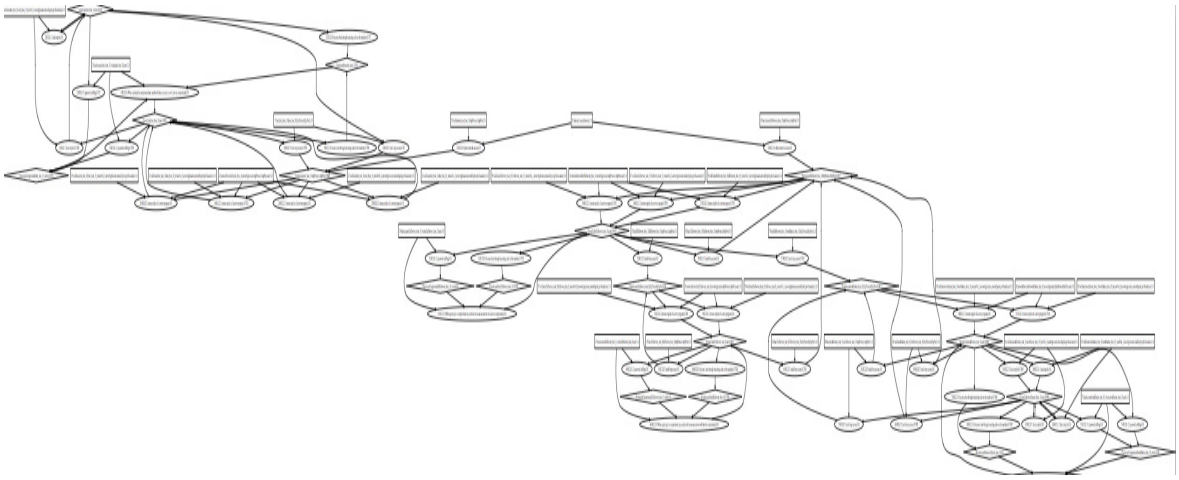
Figure 5.5: Chain attack network connectivity

difficulty if they have a probability of 0.0-0.39. Scenarios will be labeled “Medium” difficulty if they have a probability of 0.4-0.69. Scenarios will be labeled “Low” difficulty if they have a probability of 0.7-1.0.

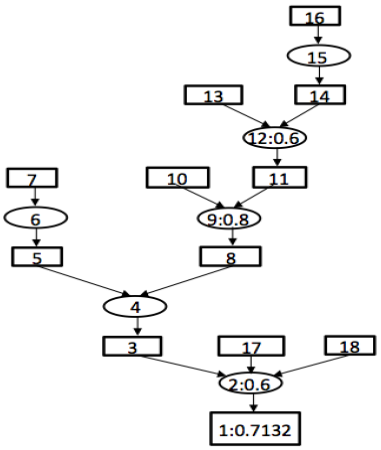
5.4 Evaluation

We have implemented our algorithm in the Python language and evaluated its effectiveness using a chain attack scenario to gain empirical experience of the metric model. As shown in Figure 5.5, the scenario contains two interconnected subnets. The objective of the exercise is to simulate multi-stage attacks. There are ten hosts in this scenario. Hosts 1 through 5 belong to subnet-1; hosts 7 through 10 belongs to subnet-2. Host 6 is a router that connects to both subnets.The network is pre-configured such that hosts 7 and 9 have the following vulnerabilities: application vulnerability, authentication leak, web server vulnerability, improper access control vulnerability.

By putting all the configuration files into our algorithm, we generate the attack graph with probability corresponding to the chain attack scenario as shown in Figure 5.6. Figure 5.6(a) shows the originally generated MulVAL attack graph with cycles and Figure 5.6(b) presents the trimmed attack graph with probabilities after applying our algorithm. The probability of achieving the attack goal is 0.7132, which indicates this scenario has low difficulty

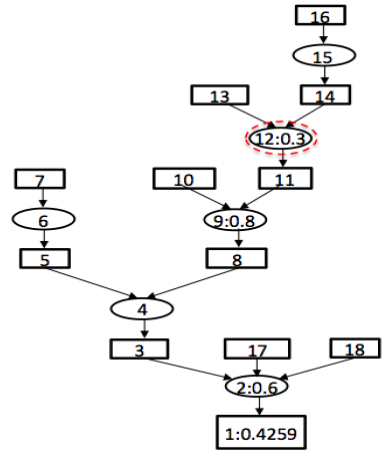


a



1: shutDownApplication('Node1','Node9','CBR')
2: remote exploit of a server program
3: netAccess('Node1','Node9',tcp,80)
4: multi-hop network access
5: netAccess('Node7','Node9',tcp,80)
6: RULE 0 (direct network access)
7: hacl('Node7','Node9',tcp,80)
8: gainRootAccess('Node1','Node7',root)
9: password sniffing
10: vulExists('Node7',' Authentication_Leak')
11: gainAccessControl('Node1','Node7'
12: remote exploit of a host vulnerability
13: vulExists('Node7','Improper_Access_Control')
14: netAccess('Node1','Node7',tcp,80)
15: direct network access
16: hacl('Node1','Node7',tcp,80)
17: vulExists('Node9',' Application_Vulnerability', 'CBR',remoteExploit,shutdown)
18: attackerLocated('Node1')

b



1: shutDownApplication('Node1','Node9','CBR')
2: remote exploit of a server program
3: netAccess('Node1','Node9',tcp,80)
4: multi-hop network access
5: netAccess('Node7','Node9',tcp,80)
6: RULE 0 (direct network access)
7: hacl('Node7','Node9',tcp,80)
8: gainRootAccess('Node1','Node7',root)
9: password sniffing
10: vulExists('Node7',' Authentication_Leak')
11: gainAccessControl('Node1','Node7'
12: remote exploit of a host vulnerability
13: vulExists('Node7','Improper_Access_Control')
14: netAccess('Node1','Node7',tcp,80)
15: direct network access
16: hacl('Node1','Node7',tcp,80)
17: vulExists('Node9',' Application_Vulnerability', 'CBR',remoteExploit,shutdown)
18: attackerLocated('Node1')

c

Figure 5.6: Chain attack graph with probabilities

level. Assume as time passes, there might be a new patch for the vulnerability on node 12. As shown in Figure 5.6(c), the node 12 probability changes from previous 0.6 to current 0.3 because it is now harder to exploit. Corresponding by, the probability of achieving the attack goal changes to 0.4259, medium probability indicate medium difficulty level.

Possible sequence of attacks launched by host 1 to victimize host 9 are as follows:

- Host 1 launches an attack on host 7 trying to exploit improper access control vulnerability. Since host 7 has this vulnerability, the attack is successful.
- Host 1 tries to exploit authentication leak vulnerability on host 7; host 7 has this vulnerability and the attack is successful.
- Host 1 tries to attack host 9 via host 7's authentication leak vulnerability, host 1 was able to gain access to host 7. Then host 1 tries to exploit application vulnerability on host 9, attacking from host 7. The attack is successful; host 1 is able to obtain information on the application(s) running on host 9.
- Host 1 attacks host 9 by shutting down its application. The attack is successful.

5.5 Conclusion

We have presented a graph-based approach to calculating an attacker's success likelihood of achieving attack goal, which can be used as a metric to evaluate the difficulty levels of training exercise scenarios. Due to the causal relationships between vulnerabilities in an attack graph, we apply Bayesian Reasoning to calculate the final probability. We confirmed the metric's effectiveness by evaluating it in empirical scenarios.

In this chapter, we design efficient algorithms to remove cycles in the attack graph while keeping their influence on the final probability calculation. Since we can differentiate the difficulty levels of penetration exercises in training system, it is desirable to devise algorithms to compute cyber defense analysts performance scores. Varied mission goals will place different priorities on security requirements, and hence different metrics and scoring algorithms. As an example, for a banking institution, it is more important to protect confidentiality and integrity of customer information, so any attacks compromising these will

be weighted higher. For a public library where the information is free to the public, it is less important to guard against confidentiality and integrity, however, more weights will be assigned to protecting the availability of its services.

Chapter 6

WEB-BASED FUZZY TEAM DECISION-MAKING FOR CYBER SITUATION AWARENESS

6.1 Introduction

Situational Awareness (SA) involves perception of evolving status and attributes of elements, comprehension of combined observations to evaluate the current situation, and prediction of possible future outcomes based on past experience and knowledge [47]. Cyber Situation Awareness (CSA) extends SA to the cyber domain, where cyber analysts collect data and seek cues to form attack tracks, estimate the impact of observed attack tracks, and anticipate the moves (actions, targets, time) of attackers.

CSA is a cognitive process of human cyber analysts. Although the cognitive work of CSA is similar in many ways to other work in complex systems, there are several features of cyber security that make CSA unique [94, 95]. First, the operational space of CSA is boundless with minimal geographical features. As a result, the environment from which a cyber analyst has to perceive salient cues is vastly larger and more difficult to comprehend. Second, the speed at which the cyberspace changes is much faster, where new vulnerabilities and their corresponding exploits are continuously emerging, and new offensive technologies are constantly being developed. Furthermore, modern exploits are either employed via misdirection (e.g., a DDoS attack is conducted by a botnet of compromised computers) or delivered passively via embedded malware. Third, everything cyber analysts know about the environment is a virtual representation of the cyberspace in terms of digital information (e.g., intrusion alerts and firewall logs). In addition, the cyber analyst only sees the information that his/her (software) sensors are capable of detecting in a form that can be rendered on monitor screen.

Because perception and comprehension of cyberspace is inherently constrained by technology artifacts, cyber analysts' ability to develop situation awareness is greatly limited by the degree to which the network's sensors are correctly configured and capturing data.

When confronted with sheer amount of situation information and dynamically changing environments, cyber analysts, at different levels and/or in different parts of the system, need to work collaboratively as a team. Typically, each team member forms his/her own CSA and shares it with other team members in order to create team CSA. However, each individual cyber analyst may have his/her own personal expertise, experience, and opinions, so that conflicts often occur in team decision making on, for instance, whether there exists a cyber attack, or it is a false alarm or not, as well as the types of detected cyber attacks. Therefore, how to resolve conflicts among team members becomes a critical issue. Observation from the previous "Capture The Flag" cyber competitions show that traditional methods to achieve consensus decision, such as verbal discussion and whiteboard session, are not accurate and unpersuasive.

In this chapter, we investigate a fuzzy set [22] based approach that can quantify cyber analysts' preference and aggregate uncertain information to generate consensus of CSA for the entire team. Considering the collaboration among team cyber analysts may be across physical distances and need web-based technology to facilitate, we also design a web-based team CSA decision-making system based on the fuzzy set method.

The remainder of the chapter is organized as follows: Section 6.2 describes the background, the related work, and the motivation for utilizing fuzzy set method to process uncertain information. Section 6.3 describes the designed methodology for team CSA, which includes team structure and roles of cyber analysts, communication among cyber analysts, individual CSA representation, and fuzzy set based decision support for team CSA. In Section 6.4, we present the web-based team CSA decision-making system that utilizes the fuzzy set approach. Section 6.5 concludes the chapter.

6.2 Background and Related Work

Whenever a security incident occurs, the top three questions that any cyber analyst will ask are: what had happened, why did it happen, and what should I do. Answers to these questions lie in the core of Cyber Situation Awareness [27, 55, 69], which is a cognitive recognition and realization process of identifying critical security information in the environment. It has long been acknowledged that CSA or the ability to assess situations and prepare timely responses is an important aspect for cyber security defensive purposes.

According to the Situation Awareness (SA) reference model proposed by Endsley [25], SA is a three phases process: perception, comprehension, and projection. SA begins with perception that provides information about the status, attributes, and dynamics of relevant elements within the environment. Perception also includes classifying information into understood representations and provides the basic building blocks for comprehension and projection. Comprehension of the situation encompasses how people combine, interpret, and correlate information, which yields an organized picture of the current situation by determining the significance of objects and events. Furthermore, as a dynamic process, comprehension must combine new information with already existing knowledge to produce a composite picture of the situation as it evolves, which is projection. Cyber Situation Awareness (CSA) is SA extended to the cyber domain. Similarly, CSA also has three phases: collecting data, estimating the impact of observed cyber attacks, and anticipating the next moves.

A team is defined as a group of heterogeneous people working together towards a common goal. The heterogeneity could be based on their individual skill, information they know, or the resources they have. Team CSA is defined in [12] as the degree to which every team member possesses the CSA required for his or her responsibilities. The team members through team interactions transform individual knowledge to collective knowledge and in the process achieve team situation awareness [12, 16]. However, team situation awareness is more than the sum of situation awareness of the individuals in the team [77]. Since each team member may have his/her own personal expertise knowledge, experience, and opinions, it is hard for them to make consensus decision. Besides, cyber analysts often describe the

situation with imprecise or ambiguous information, which exacerbates the uncertainty of shared situation awareness.

CyberCog [76] is a synthetic task environment for understanding and measuring team situation awareness, as well as for evaluating algorithms intended to improve cyber situation awareness. CyberCog provides an interactive environment for conducting human-in-the-loop experiment in which the participants of the experiment perform the tasks of cyber analysts in response to cyber attack scenarios. CyberCog utilizes a collection of known cyber defense incidents and analysis data to build a synthetic task environment. Alerts and cues are generated based on emulation of real-world analyst knowledge. From the mix of alerts and cues, cyber analysts have to identify threats and vulnerabilities as a team.

In the scenarios developed by CyberCog, cyber analysts work together as a team. For instance, each cyber analyst receives individualized training on his/her specific role, such as Malware specialist, Denial of Service specialist or Phishing attack specialist. During training, if one cyber analyst encounters alerts that he/she is not very familiar with, he/she can share the alerts with the rest of the team to ask other cyber analysts for help. CyberCog provides a team collaboration tool through which team members could share event information to get help with unfamiliar event patterns. Other team members may reply to a shared event with details and information on what needs to be done and how to carry out an investigation process for this event pattern. This interaction is very similar to the interaction patterns among cyber analysts in the real world. However, one major limitation of CyberCog's team CSA is that it is achieved via verbal discussion and lacks a quantification method to aggregate individual CSA to create team consensus.

6.3 Team Collaboration for Cyber Situation Awareness

Team CSA typically includes collaboration, information sharing or communication, and CSA aggregation/integration among team cyber analysts. To develop more effective team CSA, three main issues should be considered: (1) Uncertain information of CSA: in real world scenarios, after collecting related data, human cyber analysts tend to use imprecise or ambiguous terms to present it, for example, 'extremely large', 'very high' and 'very low'

[99, 24]. (2) Inconsistent CSA among team members: each individual cyber analyst may have his/her own personal expertise, preference, and opinions, so that conflicts may occur during team decision making. (3) Distributed operation environment: when team cyber analysts collaborate in a face-to-face environment, it is easy for them to share information between each other. However, in real word scenarios, cyber analysts may be located in different physical locations or be involved in different parts of the system. In this case, communication is the key for team CSA to reduce the occurrence of discrepancies in shared CSA.

In this section, we introduce the designed methodology for team CSA to address these identified issues, which includes: team structure and roles for cyber analysts, communication among cyber analysts, individual CSA representation, and fuzzy set based decision-making for Team CSA.

6.3.1 Team Structure and Roles for Cyber Analysts

A team of interdependent cyber analysts is not able to function well without a clear team structure or well assigned roles in a team. There are some critical questions such as "who knows what? who takes what action? who needs to know this? " that should be answered. However, team structure and roles may be inconsistent across organizations. In CyberCog, each team member is assigned with an attack-based role, such as Malware specialist, Denial of Service specialist and Phishing attack specialist, etc. In [20], the authors concluded through Cognitive Task Analysis (CTA) for the US Air Force that there are six roles of the cyber analysis task: triage, escalation analysis, correlation analysis, threat analysis, incident response, and forensic analysis. A study conducted by the U.S. Government Accountability Office concluded that there were many conflicts regarding role positions, responsibilities and implementations across organizations. For our system, team members organize themselves and split work in an ad hoc manner while responding to attacks. All team members are trained for recognizing all types of cyber attacks. After the training, a performance test is given to examine their proficiency in identifying different types of cyber

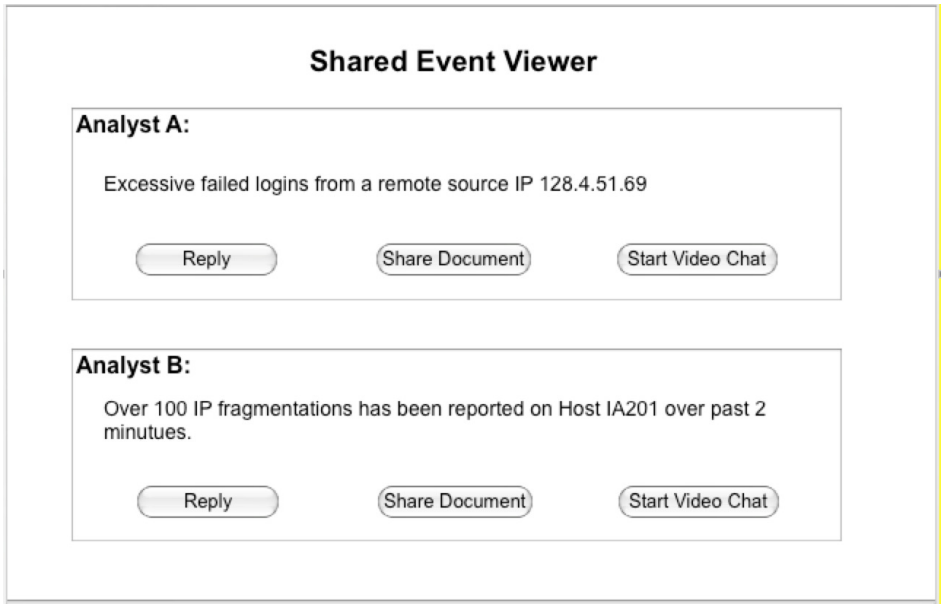


Figure 6.1: Team Communication Screen Example

attacks. Such individual performance will be considered as the 'weight' of their opinions on the following team decision-making process.

6.3.2 Communication among Cyber Analysts

Previous observations of the Air Force Academy's participation in the Cyber Defense Exercise (CDX [4, 85]) concluded that a team of cyber analysts does not work well as expected. One major issue is that cyber analysts tend to work independently without enough communication among them. In addition, the competition between team members often lead to 'silence' in order to appear more prestigious if they were the ones to report a security breach alone. Analysts also reported that they occasionally worked on the same data set as other analysts, and only realized the fact after the task was completed. Therefore, team members with distinct expertise/knowledge should help one another to correctly identify attacks so as to enhance the whole team's performance. As depicted in Figure 6.1, cyber analyst can post their suspicious events and other team members can discuss them through video chat, file sharing, or whiteboard among themselves.

6.3.3 Representation of Individual CSA

Usually, cyber analysts detect incoming attacks by collecting and analyzing network traffic and suspicious or unexpected behavior. Such information is represented as events combining (1) alerts specified by cyber analysts when watch list items exceed certain triggering thresholds and (2) alerts from Intrusion Detection System (IDS), such as snort, virus scanning software reports, and operating system, service and application logs.

Typically, cyber analysts tend to transform observed events into their own cyber situation awareness with descriptions such as follows:

- High memory usage on host A
- Very low CPU utilization on host B
- Unusually large data uploads from host C
- Excessive failed logins from a remote source IP
- Host D receives UDP packets with extremely large payloads
- The large number of IP fragmentations on Host E indicates a potential DoS IP Fragmentation Attack
- From my experience, it might be Wireless Jamming Attack

Notice that human cyber analysts are often only imprecisely or ambiguously aware of a situation so that uncertain terms, such as 'excessive', 'high' , 'very low' and 'my experience' are used to quantify such situation. As team CSA has to be created through aggregating such imprecise information and opinions, fuzzy set [22] based decision-making approach becomes suitable to quantify cyber analysts' opinions and generate team CSA.

6.3.4 Fuzzy Set based Decision-Making for Team CSA

We use fuzzy set based method to achieve team CSA by letting team members make consensus decisions on the types of cyber attacks. Fuzzy set has been applied in the area

of multi-criteria team decision-making to address the uncertainty issue in generating consensus opinions, such as the problem of facility location selection [53, 36, 9]. Such method constructs preference relation between solution alternatives by evaluating different criteria, and selects the best solution from a set of alternatives that is most acceptable by the entire team. Specifically, we use fuzzy set to facilitate a team of cyber analysts to make consensus decision when they have conflicting judgements on the types of on-going attacks. First, we present the notations used.

Let $P = \{P_1, P_2, ..., P_n\}, n \geq 2$, be a finite set of at least two decision makers (cyber analysts) to select satisfactory solutions from alternatives; $S = \{S_1, S_2, S_3, ..., S_m\}, m \geq 3$, be a finite set of at least three solution alternatives for a given decision problem; $C = \{C_1, C_2, C_3, ..., C_t\}, t \geq 2$ be a finite set of at least two selection criteria for the solution alternatives. The procedure of fuzzy set based team decision making consists of eight steps as shown in Figure 6.2.

Step 1: Determine the solution alternatives

When a decision problem of identifying the type of an on-going cyber attack is presented to a CSA team, cyber analysts may propose different alternatives regarding the type of the on-going cyber attack. After collecting all the possibilities, the alternatives set is defined as $S = \{S_1, S_2, S_3, ..., S_m\}, m \geq 3$.

Step 2: Choose selection criteria

Each team member can also propose several solution criteria for assessing solution alternatives. Criteria proposed by all the team members are put into a criteria pool. If the criteria pool becomes too big, only the top-t criteria, $C = \{C_1, C_2, C_3, ..., C_t\}$, will be chosen through voting for the purpose of computational efficiency.

Step 3: Determine the weights of decision makers

Based on their individual ability, experience, and other factors, team members may have different degrees of influence on decision-making, and hence should be assigned different 'weights'. Each cyber analyst will be evaluated to determine his/her own 'weight'

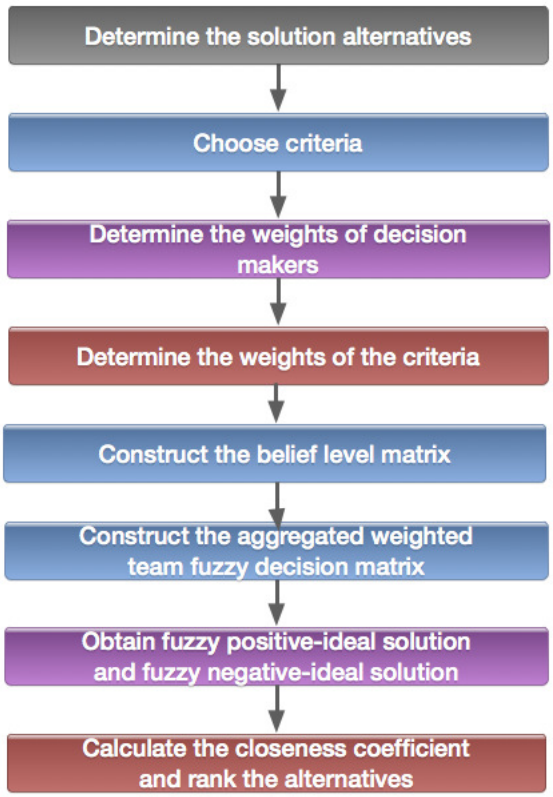


Figure 6.2: The procedure of the team decision-making

based on his/her ability, experience, etc., which determines his/her his/her influence on decision making. Example weights assigned to cyber analysts are presented in Table 6.1. The normalized weights of all the decision makers from the *normalized decision maker weight vector* v as follows.

$$v = (v_1, v_2, ..., v_n) \text{ and } \sum_{k=1}^{n} v_k = 1 \tag{6.1}$$

Table 6.1: Weight of individual cyber analyst

Individual Performance	**Weight**
Excellent	4.0
Great	3.7
Good	3.3
Normal	3.0
Not Good	2.0

Step 4: Determine the weights of selection criteria

In the process of decision making, decision makers are required to express their preferences on selection criteria through pairwise comparison. The linguistic terms of comparison between selection criteria are shown in Table 6.2. The comparison scale ranges from 1 to 9, representing the concepts of: 1 - equally important; 3 - slightly more important; 5 - more important; 7 - strongly more important; 9 - absolutely more important. Values 2, 4, 6 and 8 are intermediate values between adjacent judgments. We adopt the pairwise comparison method of the Analytic Hierarchy Process (AHP) [79]. For decision maker, the pairwise comparison of t criteria generates a $t \times t$ criteria matrix w^k, where entry w_{ij} represents the importance of the ith criterion with respect to that of the jth criterion. If $w_{ij} > 1$, the ith criterion is more important than the jth criterion, while if $w_{ij} < 1$, the ith criterion is less important than the jth criterion. If two criteria are equally important, entry w_{ij} is 1. The entries w_{ij} and w_{ji} satisfy the relation of $w_{ij} \cdot w_{ji} = 1$.

After the initial pairwise comparison, the weight for every criterion i can be determined by calculating the geometric mean[1] of the ith row in the criteria comparison matrix. The normalized criteria weight vector w^k of decision maker P_k is denoted as:

$$w^k = (w_1^k, w_2^k, w_3^k, ..., w_t^k) \text{ and } \sum_{i=1}^{t} w_i^k = 1 \tag{6.2}$$

Table 6.2: Linguistic terms for the comparison of criteria

Linguistic terms	**Comparison Scale**
Equally important	1
Slightly more important	3
More important	5
Strongly more important	7
Absolutely more important	9

Step 5: Construct the belief level matrix

Given selection criterion C_i $(i = 1, 2, .., t)$, *belief value* b_{ij}^k specifies the preference of choosing solution alternative S_j $(j = 1, 2, .., m)$ by decision maker P_k $(k = 1, 2, .., n)$. $[b_{ij}^k]$ forms the $t \times m$ belief matrix for decision maker P_k. Such belief values refer to a set of linguistic terms denoting various degrees of preference specified by the decision makers, as shown in Table 6.3, and the normalized belief values simply reflect decision makers' preferences on choosing solution alternations. Notice that the belief value corresponding to linguistic term 'Highest,' for instance, does not have to be 0.36, as long as it is larger than the value of 'High,' and similarly for other terms.

For each decision maker P_k, elements b_{ij}^k in the jth column of its belief level matrix can be aggregated into b_j^k by multiplying them with the corresponding normalized criteria weight vector w^k, which stands for decision maker P_k's belief on the jth solution alternative

[1] The geometric mean of a group of n numbers indicates their central tendency or typical value by calculating the nth root of the product of these n numbers.

Table 6.3: Linguistic terms for belief values on solution alternatives

Linguistic terms	**Belief Values**
Highest	0.36
High	0.28
Medium	0.20
Low	0.12
Lowest	0.04

as:

$$b_j^k = \sum_{i=1}^{t} w_i^k * b_{ij}^k \tag{6.3}$$

$$j = 1, 2, ..., m; k = 1, 2, ..., n$$

The belief vector b^k for decision maker P_k is expressed as:

$$b^k = (b_1^k, b_2^k, b_3^k, ..., b_m^k) \tag{6.4}$$

Step 6: Construct the aggregated weighted team fuzzy decision matrix

Using the normalized decision maker weight vector v and the belief matrix b_j^k, we can construct a weighted fuzzy decision vector.

$$(r_1^{'}, r_2^{'}, ..., r_m^{'}) = (v_1, v_2, ..., v_n) \begin{pmatrix} b_1^1 & b_2^1 & \cdots & b_m^1 \\ b_1^2 & b_2^2 & \cdots & b_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1^n & b_2^n & \cdots & b_m^n \end{pmatrix} \tag{6.5}$$

The weighted normalized fuzzy decision vector is denoted as $r = (r_1, r_2, ..., r_m)$ and each element r_j is normazlied as:

$$r_j = \frac{r_j^{'}}{\sum_{j=1}^{m} r_j^{'}} \tag{6.6}$$

Step 7: Obtain fuzzy positive-ideal solution and fuzzy negative-ideal solution

The basic principle is that the chosen solution alternative should have the shortest distance from the positive-ideal solution and the longest distance form the negative-ideal solution. In the weighted normalized fuzzy decision vector, each r_j belongs to the close interval $[0, 1]$. We then define a fuzzy positive-ideal solution r^+ to be 1 and a fuzzy negative-ideal solution r^- to be 0. The distance between r_j and the positive-ideal r^+ toward d_j^+ and the distance between r_j and the negative-ideal r^- toward d_j^- are calculated as:

$$d_j^+ = d(r_j, r^+),\ j = 1, 2, ..., m \tag{6.7}$$

$$d_j^- = d(r_j, r^-),\ j = 1, 2, ..., m \tag{6.8}$$

where $d(\cdot, \cdot)$ is the difference between two fuzzy numbers.

Step 8: Calculate the closeness coefficient and rank the alternatives

A closeness coefficient is defined to determine the ranking order of all the solution alternatives once d_j^+ and d_j^- of each solution alternative S_j $(j = 1, 2, .., m)$ are obtained. The closeness coefficient of each solution is calculated as:

$$CC_j = (d_j^+ + (1 - d_j^-))/2,\ j = 1, 2, ..., m \tag{6.9}$$

The alternative S_j that corresponds to $argmax_i(CC_i)$ is the most acceptable solution for the team decision.

6.4 Web-based Team CSA Support System Experiment

Considering collaboration among team cyber analysts may take place in a distributed manner, we design a web-based team CSA support system based on the proposed fuzzy set method. We use the following example to illustrate the operations of the web-based team CSA support system.

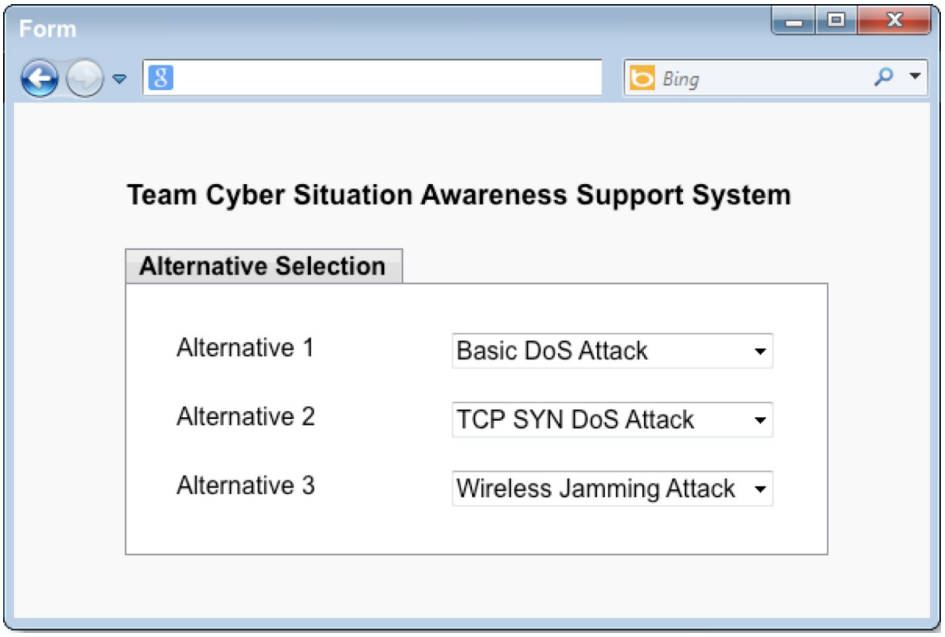


Figure 6.3: Three attack type alternatives proposed by team members

Suppose a distributed team of three cyber analysts are collaboratively observing an enterprise network. Each cyber analyst collects data and observes the different system information. When team members have conflicting judgment about the type of current cyber attack, they can login to the system and set up the assessment environment.

In the first step, all the team members submit their own suggestion of possible alternative types of cyber attacks. As shown in Figure 6.3, three cyber attack types: Basic DoS Attack, TCP SYN DoS Attack, and Wireless Jamming Attack are suggested as possible attack types.

Next, the process of choosing criteria can be done via team discussion or voting among team members as shown in Figure 6.4. Each team member proposes some criteria for assessing these cyber attack alternatives. Some cyber analysts think CPU and memory usage metrics should be considered in judging the current cyber attack type. Others believe packet send ratio is more important. Criteria proposed by all the team members are put into a criteria pool. If the criteria pool becomes too big, only the top-t criteria are chosen for the purpose of computational efficiency.

Based on the criteria proposed, each team member completes a pairwise comparison

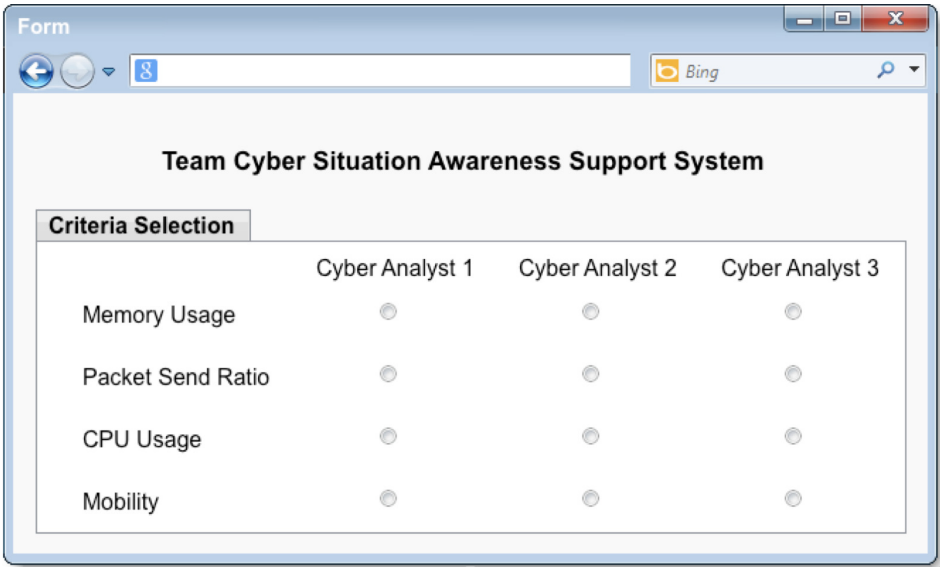


Figure 6.4: Solution criteria selection system

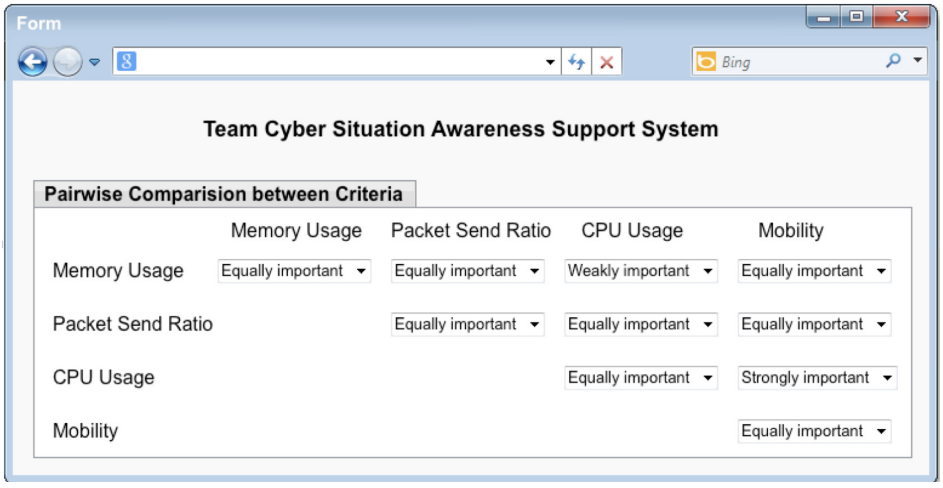


Figure 6.5: Criteria comparison matrix

to choose the relative importance of these four selected criteria as shown in Figure 6.5. By pairwise-comparing the relative importance of these selection criteria, the criteria comparison matrix $W^k = [w_{ij}^k]$ for decision maker P_k $(k = 1, 2, ..., n)$ is generated. The example criteria comparison matrix for decision maker P_1 corresponding to Figure 6.5 is as follows:

$$W^1 = \begin{bmatrix} 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 1 \\ 1/3 & 1 & 1 & 5 \\ 1 & 1 & 1/5 & 1 \end{bmatrix}$$

To generate consistent weights for all the selection criteria, the geometric mean of each row of the criteria comparison matrix is calculated and then the results are normalized.

$$\begin{pmatrix} (1 \times 1 \times 3 \times 1)^{1/4} \\ (1 \times 1 \times 1 \times 1)^{1/4} \\ (1/3 \times 1 \times 1 \times 5)^{1/4} \\ (1 \times 1 \times 1/5 \times 1)^{1/4} \end{pmatrix} \Rightarrow \begin{pmatrix} 1.31607 \\ 1.0 \\ 1.13622 \\ 0.66874 \end{pmatrix} \Rightarrow \begin{pmatrix} 1.31607/4.12103 \\ 1.0/4.12103 \\ 1.13622/4.12103 \\ 0.66874/4.12103 \end{pmatrix} \Rightarrow \begin{pmatrix} 0.3194 \\ 0.2427 \\ 0.2757 \\ 0.1622 \end{pmatrix}$$

In the GUI depicted in Figure 6.6, each cyber analyst fills up a belief level matrix to express his/her preference to the three solution alternatives under the four selected criteria. For instance, one cyber analyst thinks for the Basic DoS attack, Memory Usage metric is more important than the Packet Send Ratio metric as the Memory Usage is marked with a 'High' weight and Packet Send Ratio is given a 'Low' weight.

By combining all the matrices and calculating through Equation (5), the closeness coefficient of each alternative can be calculated. As shown in Figure 6.7, based on the calculated coefficient values, the consensus team CSA decision is that the on-going attack is a TCP SYN DoS Attack since it has the maximum coefficient value of 0.617272.

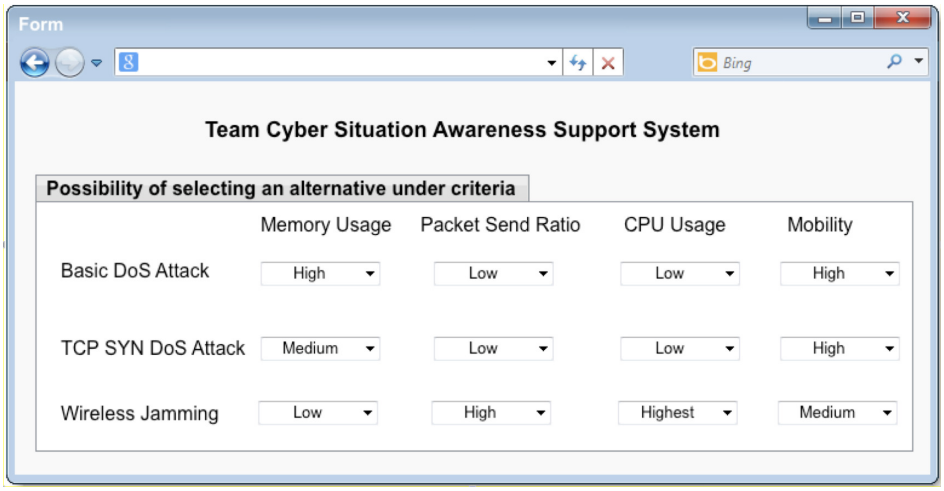


Figure 6.6: Belief level matrix filled by one cyber analyst

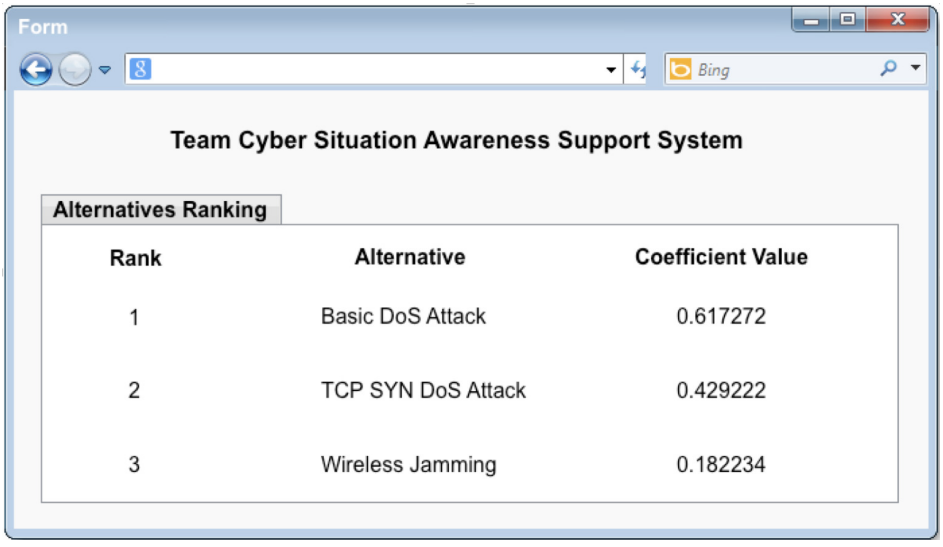


Figure 6.7: Alternatives rank based on coefficient value

6.5 Conclusion

Team CSA is a complex process requiring cyber analysts to work collaboratively at different levels and in different parts of the system. By taking into account three major issues that affect team CSA, CSA uncertain information, inconsistent CSA between team members, and distributed operational environment, we propose a fuzzy set based method to aggregate individual CSA into team CSA so as to make consensus decisions on the types of on-going cyber attacks. Besides, we design a web-based team CSA support system to demonstrate how the proposed approach can provide support for generating team CSA in a distributed manner with the ability of handling uncertain information.

Chapter 7

REALTIME CONTEXT-BASED INFORMATION FUSION FOR ADVANCED PERSISTENT THREATS INVESTIGATION

7.1 Introduction

Cyber attacks, which refer to any computer-to-computer attacks that undermine the confidentiality, integrity, or availability of a computer or information resident on it, have increased significantly in number and in complexity in recent years. Typically, a cyber attacker first exploits a system's vulnerabilities and infiltrates its network and/or hosts. Once the attacker gains entrance into the system, he may use it to monitor communications, steal critical data, discover new avenues of attack in related systems, take control of assets managed by the system or disable vulnerable networks, computers, and associated systems. Harmful outcomes of a successful attack include the attacker's ability to access sensitive data on the network and to control the hosts and network resources.

Cyber attack and anomaly detection techniques suffer from their reliance on predefined signatures of known malicious events or unusual conditions that warrant further investigation. The use of signature-based detection cannot effectively eliminate false negatives when dealing with Advanced Persistent Threats (APTs) since the financial resources and time available to APTs allows them to hide for a prolonged period and to use previously unknown "zero-day" attack. Furthermore, reliance upon signatures may unintentionally assist APTs in avoiding detection if the criteria for detection are well known.

In this chapter, to address this critical need of enabling analysts to rapidly and accurately detect APTs attacks, we design a novel non-signature based APTs investigation system that allows the proper identification, prioritization, and understanding of APTs attacks. The key innovation of our approach is to design and develop host-and network based purpose sensors and places them within the network and individual hosts to provide real-time data and

then use this information combined with network-specific knowledge to create a dynamic set of event threads that, when touched by a given alert received from traditional intrusion detection systems (IDSs), will immediately identify the context surrounding the alert and thus the automatic calculation of the alert's legitimacy and severity [39]. The result is that much of the follow-up investigation of each alert is shifted into the prioritization process that utilizes the gathered context for correctly prioritizing the analyzed alerts. The burden on cyber analyst is largely reduced both by significantly improved prioritization and by providing a contextual picture of each identified potential attack. Hence, APTs attacks can potentially be detected while still at a preparation stage and with an operationally relevant level of accuracy.

The remainder of the chapter is organized as follows: Section 7.2 describes background and related work. Section 7.3 reviews the Advanced Persistent Threats characteristics and countermeasures. Section 7.4 introduces the designed Advanced Persistent Threats Investigation System Using Purpose and Network Correlators including the System Architecture and Information Flow, Run-time Data Collection and Monitoring, Data Pre-processing and Normalization, and APTs Threat Analytic. In Section 7.5, we implemented and deployed an APTs investigation system on a cloud-based testbed and have tested its performance with three different types of APTs behaviors. Finally, Section 7.6 concludes the chapter.

7.2 Background and Related Work

Human cyber analysts have to examine huge amounts of data such as system logs, configurations, traffic logs, IDS logs, and audit logs in order to identify potential threats. Thus, they would likely be soon overwhelmed by tremendous data and forced to ignore potentially significant evidence introducing errors in the detection process. Information fusion is considered to be an effect technique that obtains more relevant and qualitatively superior information out of the huge amount of data produced by various sources, thus emphasizing the information reduction properties of information fusion, and the relative improvement in obtained information quality.

We believe that alert reduction is critical to cyber analysts in four ways: (a) data may be collected from multiple sources. The computer network may consist of interconnected

routers, switches, and heterogeneous hosts with different hardware platforms, operating systems and application software; (b) relevant data are presented at multiple abstraction levels, such as Ethernet protocol, IP, TCP, HTTP, Application layer, etc. (c) data have to be analyzed and correlated to form meaningful information, and (d) this information needs to be represented at the human level of understanding.

General reviews of current alert correlation algorithms and techniques are given in [80, 65, 82]. Alert correlation algorithms can be separated into two major categories based on their characteristics: similarity-based and knowledge-based algorithms. In [96] the author proposed that the similarity between alerts can be estimated based on similarity between respective alert attributes such as IP, port, time, attack class, etc. Similarity between alert types are defined in a static matrix which express the probability of an alert of type x be followed by an alert of type y. Klaus Julisch proposes an alert root cause analysis method [52]. The author observed that over 90% of the alerts that the intrusion detection system generated are caused by a few dozens of persistent root causes. The main idea of the proposed clustering technique is establishing the hierarchy structures, which decompose the attributes of the alerts from the most general values to the most specific. Furthermore, statistical-based techniques have been used in [19, 111] to estimate the similarity between alert pairs.

Knowledge-based alert correlation algorithms are usually based on attack scenarios' pre-requisite and possible occurring results. Then, each incident is chained with other incidents to form the attacking paths. Several researchers [109, 67, 10, 92] have proposed mechanisms of identifying causal relationships between alerts by chaining pre-requisite and post conditions. Although some knowledge-based algorithms claim to be able to correlate alerts in real-time, they heavily rely on network scenario knowledge since they have to model each network state's pre and post conditions in advance.

In the Cyber Situation Awareness reference model [47], a Joint Directors of Laboratories (JDL [55, 69]) Data Fusion model is adopted in different CSA levels. JDL Level 1 deals only with the tracking and identification of individual objects, while JDL Level 2 is the aggregation of the objects into groups or units through the identification of relationships between the objects. Although JDL has been accepted in several SA frameworks, it is

not suitable in cyber domain. The JDL method does not seriously take into account context information such as acquisition time, location, source and destination of communication, and service. However, all these factors are critical for correctly detecting threats. Take data acquisition time as an example, assuming cyber analysts detect multiple TCP connection requests on the same port 22, but if they are directed to different machines in a very short time, they may assume that an attacker is trying to check whether the SSH server is active and finding the attack entrance. On the other hand, if the port probes happen once a week, it might be just the system's probing behavior.

7.3 Review of Advanced Persistent Threats Characteristics and Countermeasure

Advanced Persistent Threats (APTs [15, 45]) are stealth attacks that gain access to defense, financial, and other targeted information from governments, corporations, and individuals. Unlike other cyber attacks aiming at immediate financial gain or information leaks, APTs require a high degree of stealth over a longer duration and maintain a foothold in these environments for future use and control. These APTs' characteristics make them difficult to be detected by the current tool such as Intrusion Detection System. Several researchers have proposed APTs investigation mechanisms [91, 81, 29], nevertheless, most of them focus more on the APTs traffic pattern rather than holistic APTs investigation mechanism.

An advanced persistent threat (APT) is a targeted intrusion effort that uses multiple attack steps to break into a network or system over a prolonged duration of operation. Compared with other existing cyber attacks, APTs are more severe due to the multi-stage (and silent) intrusions that are usually launched by determined, well-funded adversaries. In order to obtain valuable information while avoiding detection, APTs require a high degree of stealthiness over a prolonged duration, which makes it hard for them to be detected by traditional intrusion detection systems (IDSs).

Briefly speaking, APTs have the following properties:

- Advanced: Attackers can utilize the full spectrum of computer intrusion technologies and techniques. They may combine multiple attack methodologies and tools in order to reach and compromise their target.

- Persistent: Attackers give priority to a specific task, rather than opportunistically seeking immediate gain. Through a low-and-slow approach, APTs attacks continuously monitor a computer network or system and its interactions to achieve the defined objectives.

- Threat: There is a level of coordinated human involvement in the attack, rather than a mindless and automated piece of code. Attackers usually have a specific objective, and are skilled, motivated, organized and well-funded in most cases.

APTs attacks can penetrate enterprises through a wide variety of vectors, without necessarily breaching perimeter security controls from an external perspective. They can leverage both "insider threat" and "trusted connection" vectors to access and compromise targeted systems. A key requirement for APTs is to remain invisible for as long as possible. Therefore, APTs technologies tend to focus on "low and slow" attacks. They attempt to stealthily propagate from one compromised host to another, without generating irregular or predictable network traffic.

Tremendous effort has been invested to ensure that legitimate operators of the systems cannot observe malicious actions. For instance, malware is a key ingredient in successful APTs operations, as it includes all of the required features and functionalities to "invisibly" infect digital systems, hide from IDSs, navigate targeted networks, capture and extract valuable data, and remain silent with covert channels for remote control.

The remote control channel is the key and foundation of APTs. Without it, attackers cannot effectively navigate to specific hosts within a target organization, or exploit and manipulate local systems to gather critical information. This also gives security analysts an opportunity to detect APTs, since their malware remains stealthy at the host level and their network activities associated with the remote control are more easily identified, contained, and disrupted at the network level. The detail analysis of recent sophisticated Advanced Persistent Threats is described in Appendix B.

7.3.1 Life Cycle of Advanced Persistent Threats

Typically, the APTs lifecycle or its infiltration path consists of the following major phases:

1. Reconnaissance: Attackers search for points of entry, vulnerabilities, key individuals and critical assets.

2. Launch: Attackers attempt to gain access to a privileged host, using targeted attacks or spear phishing to evade detection. The most popular methods used in this attack phase include:

   - Email lures with embedded links to websites with zero-day malware downloads.
   - Emails with file attachments in common formats (e.g., Office, PDF) that include zero-day attack code.
   - Infected websites of interest to key individuals identified by social media profiles.
   - Social engineering to gain access to privileged user account credentials.

3. Infect: In this stage, malicious code (custom/zero-day attack code in most cases) is installed onto a privileged host. This code reports back to a command-and-control (C&C) location with network and other useful data to help attackers execute further attacks.

4. Control: Attackers remotely control infected hosts with a C&C service, typically on a dynamic DNS host. The C&C allows attackers to remotely update/renew malware and send commands to the compromised host. Note that in this stage commonly available toolkits are frequently used for C&C, which gives security analysts a good opportunity to detect APTs attacks at this stage.

5. Discover: Infected hosts download additional components to discover target data through the infected hosts, mapped network drives, and/or other network locations. Target data may include Active Directory (AD), certificate PKI servers, or cloud-based storage.

Common discovery methods used in this stage include: (i) monitoring data-in-use once a user accesses it with its credentials, (ii) breaking into systems where users have administration rights, (iii) discovering additional hosts within the target network, and (iv) using network or system vulnerabilities to infect other hosts. Note that most tools used in this stage are standard network tools, such as gsecdump [103], Cain and Abel (to crack passwords) [104], SSH [106], and RDP [105].

6. Persist: Unlike traditional malware (which will be removed by itself or an antivirus program once identified), an APTs is designed to go unnoticed. It can persist by calling back to C&C centers for updates or download new undetected (zero-day) code to avoid detection by updated antivirus solutions.

7. Extract: After attackers have taken control of one or more hosts within the target network, established access credentials to expand their reach, or have identified target data, they will send the collected data back to either the C&C server or a previously unused server. This phase can go on for a long time if the target data continues to become available (e.g., updated or new customer records) and holds value for the attacker.

8. Further Actions: Even for an attack that is eventually stopped (e.g., attackers achieved their goal, the victim noticed and cut off the attack), some long-term consequences can still result. For example, attackers may ask for a ransom, share or sell attack methods, sell the extracted data, or publicly disclose the theft.

7.3.2 Advanced Persistent Threats Characteristics

APTs attackers often use a variety of email-based techniques to create attacks, supported by other physical and external exploitation techniques. Typical characteristics of APTs attacks that are not found in other forms of attacks can be briefly summarized as follows:

- Recon: APTs attackers typically have reconnaissance intelligence and know something about their target. For example, who is the specific user they target? What

systems can help them achieve their goals? This information is often gathered on the first step of the life cycle of an APTs attack, through social engineering, public forums, and/or nation-state security intelligence.

- Time-to-live: APTs attackers usually employ techniques to avoid detection over a prolonged duration of operation. They attempt to clean up their trail and usually perform their functions during non-business hours, and always leave backdoors so that they can reenter the system in case their original access is detected. This allows them to remain persistent.

- Advanced Malware: APTs attackers usually have the ability to use the full spectrum of known and available intrusion techniques, and combine various methodologies to reach their goal. Many APTs attackers do use commercial malware and toolkits, but some of them also have the technology and expertise to create their own custom attack tools.

- Phishing: Most APTs employ internet-driven exploitation techniques and start with social engineering and spear-phishing. Once a target machine is compromised or network credentials are given up, the attackers can actively take steps to deploy their own tools to monitor and spread through the network as required, from machine-to-machine, and network-to-network, until they find the valuable information they are looking for.

- Active Attack: Rather than fully automatically sends back data using automated malware, human coordination is often involved in APTs attacks. These well-funded, motivated, skilled, and highly directed attackers making APTs attack approach and response extremely active.

7.3.3 Advanced Persistent Threats Security Challenges

APTs have become a menace to todays enterprises as they can circumvent traditional security safeguards. The reason why APTs attacks can avoid detection of existing security

monitoring tools and intrusion detection/prevention technologies is that APTs attacks start with convincing social engineering tactics. To penetrate a specific organization, attackers first gather intelligence on a few key employees by conducting web-based research and using social networking sites, such as Facebook and LinkedIn. Then attackers use this information to personalize their social engineering attacks in order to establish trust with a victim so they may download malicious code, open an attachment, or double click on a malicious link without questioning the source.

Endpoint security tools are not used effectively. Although enterprise endpoint systems are instrumented with security software, some are not configured for maximum security protection due to various reasons. For instance, some users may think that security protection will adversely impact their systems' performance, or they just do not have time or resources to learn how to use or update endpoint security tools effectively.

APTs exploit gaps between multiple security defenses. APTs are designed to take advantage of the limitations of islands of security within IT. For example, many enterprises use different tools for endpoint security, e-mail security, and web security, and there is no overall policy management or oversight across all of them. These kinds of security gaps can be utilized by APTs to monitor network openings and squeeze their way into the victim organizations.

APTs attempt to make malicious intrusions look like normal network behaviors to monitoring tools. Once an initial endpoint is infected, APTs will conduct multiple types of activities within the network, such as compromising other systems, harvesting user or administrator credentials, locating and copying sensitive data, and finally transferring the valuable data to their web-based servers. Since these activities are purposely hidden within typical day-to-day communications and network activities, most existing monitoring tools cannot reliably find them.

7.3.4 Advanced Persistent Threats Countermeasure Strategies

Obviously, there is no simple way to protect enterprises against APTs attacks, as they remain persistent once they penetrate a network or system. According to the 2013 Verizon

Data Breach Investigations Report, about 95% of targeted threats and APTs use some form of spear phishing as their starting point of the attack. Therefore, effective APTs defense strategies must include a detection solution that can identify targeted threats in emails. For example, security analysts can rewrite the embedded URLs in suspicious emails, and then maintain a constant watch on that URL for malicious behavior. By analyzing its unusual patterns in traffic, this approach could potentially identify/detect such APTs attacks, and analyze which users have been compromised, when, and how.

The key strategy to effectively detect APTs attacks is utilizing a combination of detection technologies that can triangulate logs and identify abnormal behaviors within an enterprise network. To apply this defense strategy, security analysts need to find (or develop) a number of specific APTs activity and/or event detection solutions that can work together (at different layers) to provide intelligence on the targets, the applied attack methods, the frequency of attack activities, the origination of the attack, and the potential risks.

In this section, we will briefly introduce some potential countermeasures against APTs attacks. They may give us a precise understanding of the state-of-the-art countermeasures and provide a firm starting point to build the designed system. Note that none of the individual countermeasures are perfect and ideal to detect most APTs. Most countermeasures are effective at combating a subset of APTs, however, further investigation is required to better utilize the countermeasures to bolster detection accuracy and the overall efficiency of APTs detection mechanisms.

7.3.4.1 Identify APTs Malware Signature

Note that APTs malware usually has the following characteristics: (i) hides in plain sight, (ii) initiates outbound network connections, (iii) avoids anomaly detection through outbound HTTP connections, process injection, and service persistence, (iv) 100% of APTs backdoors made only outbound connections, (v) 83% of APTs communications used TCP port 80 or 443, and (vi) simple malware signatures, such as MD5 hashes, filenames, and traditional anti-virus methods, usually yield a low rate of true positives.

According to these characteristics, we can identify a number of factors associated with APTs attacks:

- Sudden increases in network traffic, outbound transfers

- Unusual patterns of activity, such as large transfers of data outside normal office hours or to unusual locations

- Repeated queries to dynamic DNS names

- Unusual searches of directories and files of interest to an attacker, e.g., searches of source code repositories

- Unrecognized, large outbound files that have been compressed, encrypted password-protected

- Detection of communications to/from bogus IP addresses

- External accesses that do not use local proxies or requests containing API calls

- Unexplained changes in the configurations of platforms, routers or firewalls

- Increased volume of IDS events/alerts

7.3.4.2 Leverage the State-of-Art Open-Source Tools

As we mentioned previously, APTs attacks highly depend on remote access and control. Their network activities associated with the remote control could be identified, contained and disrupted through the analysis of internal and outbound network traffic. This kind of APTs detection techniques can be implemented and extended through existing open-source software tools. Below is a short list of some of those available open source tools:

- Snort: an open source network-based intrusion prevention and detection system (IPS/IDS) that employs signature and protocol, as well as anomaly-based inspection.

- Scapy: a packet manipulation tool that can create packets for a wide range of protocols. It can send and receive packets and match requests and replies. Scapy is extensible via Python scripts and can be used for a variety of detective measures.

- OSSEC: a host-based open-source IDS. Its correlation and analysis engine provides log analysis, file integrity checking, Windows registry monitoring, rootkit detection, and time-based alerting, as well as active response. OSSEC can support most operating systems.

- Splunk: a search, monitoring and reporting tool that integrates logs and other data from applications, servers and network devices. Splunk data repository is indexed and can be queried to create graphs, reports and alerts.

- Sguil: facilitating the practice of network security monitoring and event driven analysis using an intuitive GUI that provides access to real-time events, session data, and raw packet captures.

- Squert: a web application used to query and view event data stored in a Sguil database. It can provide additional context to events through the use of metadata, time series representations, weighted and logically grouped result sets

7.4 Realtime Context-based Information Fusion for Advanced Persistent Threats Detection

7.4.1 System Architecture and Information Flow

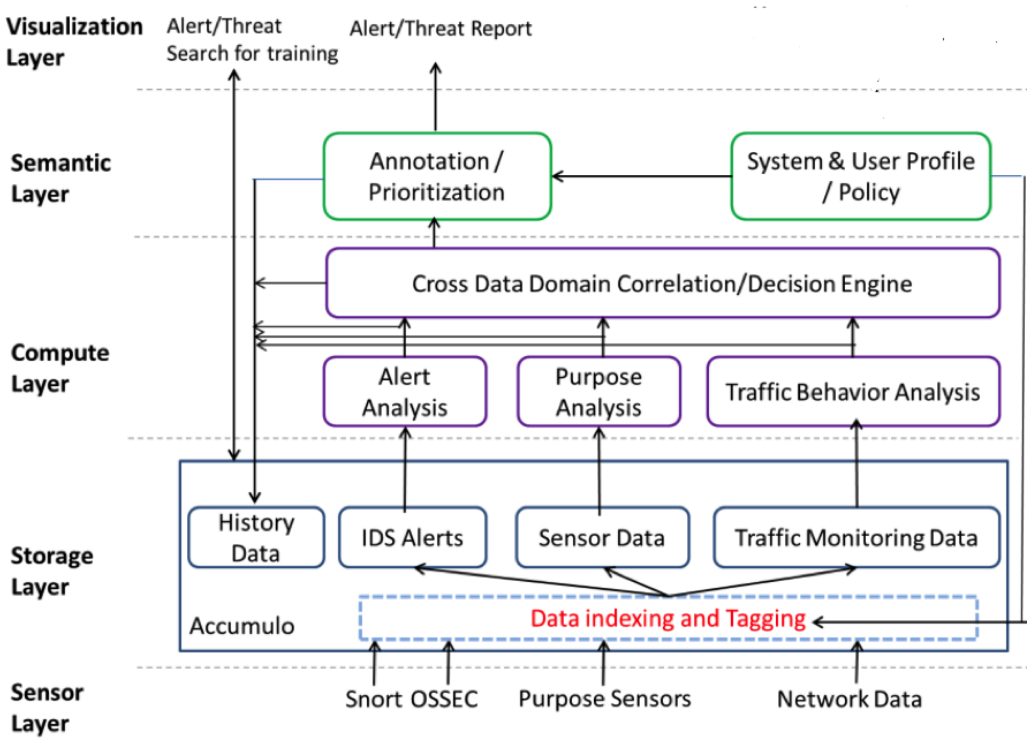


Figure 7.1: APT detection system architecture

APTs detection system is a novel signature-less system for APTs detection that allows the proper prioritization and correlation of alerts and facilitates the identification and understanding of continuous and persistent attacks. Purpose sensors are placed within the target network and individual hosts to provide real-time purpose and correlation inputs. It continually captures and analyzes attributes of potential attacks which are typically manually measured and verified only in response to highly credible alerts. Figure 7.1 shows a main diagram of the overall multi-layer system architecture that is composed of:

- Purpose Sensor - has four types of purpose sensors: (i) human interaction sensors (e.g., USB access, keyboard input, etc.) that determine which actions done by human-generated events (e.g., user initiated data transfer vs. data exfiltration), (ii) process reputation sensors that track the interactions between processes within a host, (iii) resource exposure sensors that provide information on the actions of each process, and (iv) information flow sensors that monitor which processes are communicating via the network. Based on the extended APTs behavior analysis at different stages, the following additional sensors to detect APTs at multiple stages are recommended:

  - DLP sensors use contextual relationship among data (e.g., ontology, context graph, etc.) and sensitive and monitor outbound and internal traffics to detect APTs in Stage 4, Stage 5, Stage 6 and Stage 7.

  - Behavior sensors use real-time network device monitoring at each host and log/statistics (e.g., netflow, sflow, SNMP, etc) from network device to monitor abnormal activities/traffic patterns at network and host level. These sensors are most effective for APTs in Stage 4, Stage 5, Stage 6 and Stage 7.

  - Suspicious content sensors that monitor suspicious web links/attachments in email, web traffic, and their application content to detect APTs in Stage 1, Stage 3, and Stage 3. The correlation of purpose sensor outputs with existing IDS alerts will significantly reduce false positives.

- Context Information Rather than trying to reach security decisions based on analytics working independently, it will make contextual decisions based on information

derived from multiple analytics. This context awareness is the key to detecting complex APTs that are particularly designed to evade standalone content analytics like antivirus. Specifically, it will combine real-time IDS alerts, purpose sensor outputs, and the mission-, system- and network- specific aspects of the context to generate a complete picture that is needed to correctly detect potential APTs attacks. Example context information includes: mission importance/dependencies among hosts, system/user profile and role, vulnerability vectors describing abstract paths attacks may take (e.g. the firewall allows node A to connect to node B), hosts that should be communicating, potential attack targets, etc. We believe the derived context data can significant reduce the false positives and increase APTs detection efficiency and accuracy.

- Sensitive Data Identification Data theft is a common behavior in APTs and creates artifacts for the outbound traffic stream. Users need to identify data types of interest (e.g., documents, pdf, etc.) and location/folder names (e.g., program files, windows registry, etc.). For example, we can remove data from insecure or unnecessary locations, ensure that appropriate access control and attack prevention systems are deployed in all areas where sensitive data is stored, and real-time monitor and prevent data theft at email/web gateways and critical hosts. Due to the practical considerations for providing this data, this component is optional and the system will be evaluated with and without it running.

- Multi-level Correlation When suspected attacks occur, an operator normally manually looks at each alert to determine both the cause of the attack (i.e. the purpose behind the alerts) and the true severity of an attack that may be present. However, the system correlates threat alerts at a low level. The threat analysis output is then further correlated among different types of sensor data to even further reduce false positives. For example, to detect Stuxnet [107], the system correlate network traffic analysis output with purpose sensor data. Once network data analysis detects the potential threat, it is correlated with purpose sensor output/analysis (i.e., user input cue data) to decide whether such traffic was generated by users or bots.

- Event Context Graphs In addition to providing a more appropriate prioritization of events, the information used internally by the system is delivered to the operator as part of each notification of a potential attack. This contextual knowledge provides the operator insight into why each event was deemed important, allows the operator to more quickly determine the validity and scope of the event, and gives the operator a significant head start in determining the correct response.

Figure 7.2 shows an information flow of the designed APTs Detection System. The run-time information that arrives at system in form of raw alerts from purpose sensors and third-party network analysis tools such as Intrusion Detection/Prevention Systems (IDSs/IPSs), Netflow formatted network traffic data are collected. Then, each raw alert is translated into a standardized alert format of source object, action, and sink object. Each record is assigned with a standardized name, and its attributes are copied to the appropriate fields of the alert as defined by the attribute mappings in the cloud-based database. Notice that certain important alerts generated by purpose sensors will trigger corresponding threat analytic service. The analytic service can query the database to retrieve relevant events and store the generated analysis result back into database if necessary. When triggered by certain alerts from purpose sensor, the threat analytic service is capable of generating appropriate event contexts for each attack type by applying backward tracking method. Then, using this context to assess the probability of APTs presence through inexact matching with APTs profiles.

7.4.2 Run-time Data Collection and Monitoring

The designed APTs detection system collects run-time data from the Human-Process purpose sensors, Process-Network Purpose Sensor and third-party network analysis tool such as Netflow, Snort, and OSSEC.

7.4.2.1 Human-Process Purpose Sensor

The keyboard and mouse are the basic components that link the human and the computer. We monitor keyboard and mouse events of the host to understand which program has human activity/interaction. To do this, our event sensor hooks into Windows system calls

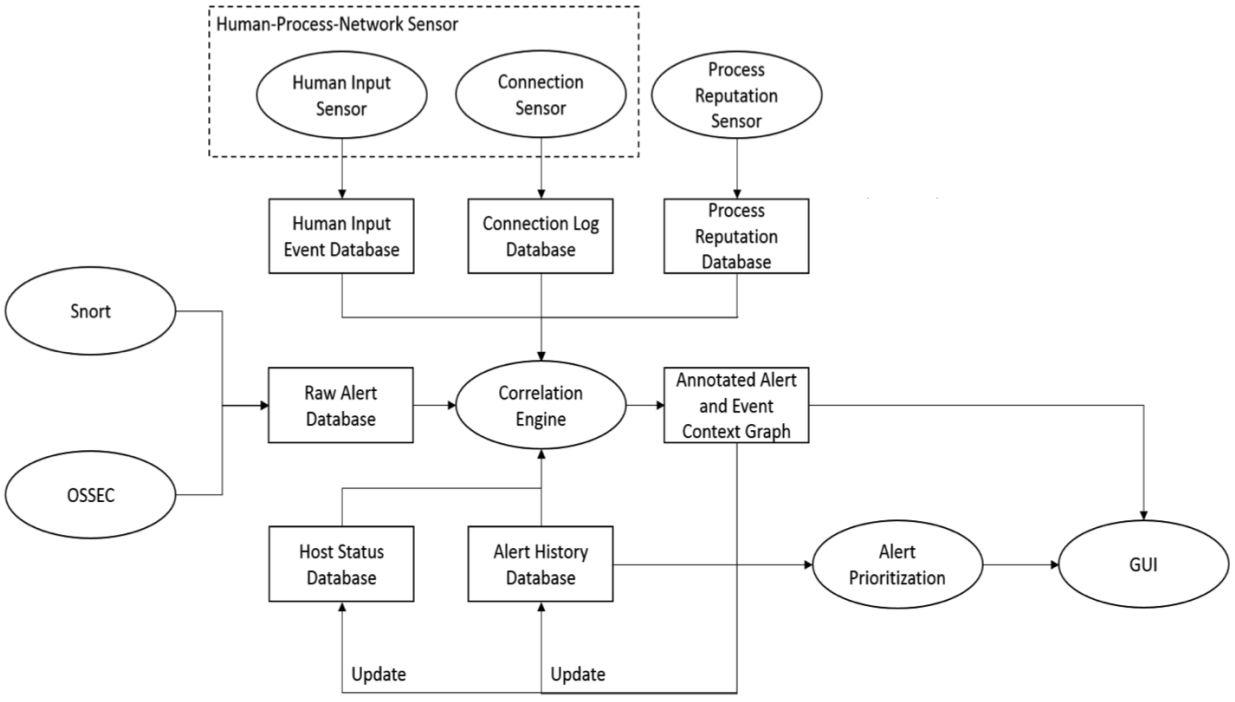


Figure 7.2: APT detection information flow

related to keyboard and mouse events and then determines which program received those events based on the current window focus.

Some previous studies [18] also apply function hooking to capture automated processes. We differentiate from that approach in several aspects. Previous approaches suffer if malware simply imitates human behaviors and creates fake mouse/keyboard events from virtual devices to confuse the sensor. To address this problem, we employ two more robust approaches. First, our sensor digs into the sources of the events. If they are from physical devices connected via the PS2 or USB interface, it trusts them; otherwise it regards them as suspicious. Second, the module investigates whether a process receiving events is running in the foreground or not. We assume that if a person legitimately produces mouse or keyboard events, a process related to the events is shown on the current screen with an activate window (i.e., running in foreground). Thus, if a process producing events is running in the foreground, we trust the process; otherwise we regard it as suspicious.

Note that in the current implementation, we trust the operating system and we believe it provides true information, a common assumption widely used in this line of research [18, 54]. Of course, some malware (e.g., a rootkit) may infect the operating system and deliver fake information, and it may even disable our sensors. This issue could be solved by employing hardware/TPM [31] or Hypervisor-based introspection and protection [5, 50].

Our human-process-network purpose sensor captures the mouse and keyboard events using Windows system functions. Basically, Windows provides functions to capture the events from external devices [64]. Using these APIs, we implement the event sensor, which identifies which process generates the events. The event sensor investigates whether the events are generated from real physical devices and checks to see if the process is running in the foreground with the help of Windows system functions. We store captured information (process, event time) to shared memory.

To capture the outgoing DNS queries, TCP SYN, and UDP packets, we use the WinPcap library to build the connection sensor. WinPcap provides functions to collect raw level network packets on the Windows OS with little overhead. Moreover, the connection sensor does not monitor all network packets, but monitors only DNS, TCP SYN and UDP packets. This approach reduces the overhead, since those packets comprise a small portion of all network packets.

Whenever there are network events to capture, the module identifies which process produces them and verifies whether the process is related to the human actions or not. However, if a process uses a helper process for a DNS query, we could not directly use it. To address this problem, we check the process that produces the DNS query and if it is a helper process (e.g., svchost.exe), the module awaits a DNS reply, which contains the IP address of the domain. Then, if there is an automatic connection from the process to that IP address after the DNS query, the module knows which process triggered the DNS query. We then use GetExtendedTcpTable and GetExtendedUdpTable functions to recognize which process created the connections. If we observe a new TCP or UDP connection, we call these functions to identify which process has the source port number of the connection.

7.4.2.2 Process-Network Purpose Sensor

Advanced Persistent Threats (APTs) typically use malware to achieve their stealthy, persistent, and advanced features. These malware programs usually work automatically without requiring human-driven activities. The purpose sensor is capable of analyzing the interaction and correlation between human activity and system processes. For example, if a computer screen capture is detected, the sensor creates an alert and searches for corresponding key strokes, otherwise it tries to find newly created or updated files that might correlate to the screen capture or newly created outgoing connection which may indicate a potential data exfiltration.

Purpose sensors also record correlated clues between system processes and outgoing network connections. They can determine whether a network connection is human-initiated or software-initiated. As a result of such correlation analysis, an alert might be generated and sent to the APTs detection system Analytics Server. For example, the sensor is used to identify the list of network connections that are not a result of human interaction on the system. It monitors network connections made by different processes, and through an analysis of mouse and keyboard events, it can detect network connections made with no human interaction and can identify that system processes that generated those connections. The module then monitors the outgoing network connections through the DNS queries issued by the correlated processes.

The overall purpose sensor components is shown in Figure 7.3, which contains two modules: M1 is System Resource Exposure Sensor and M2 is Process Reputation Sensor.

- System Resource Exposure Sensor - resides at each host to monitor the resource access activities of a suspicious process. This sensor monitors how critical resources such as files, registries, and network sockets are exposed to the target process. Currently, we utilize the SysInternals Utilities Suite [63] from Microsoft such as procmon, sysmon, and system log auditing to acquire these information.

- Process Reputation Sensor - resides at each host and consists of several modules that

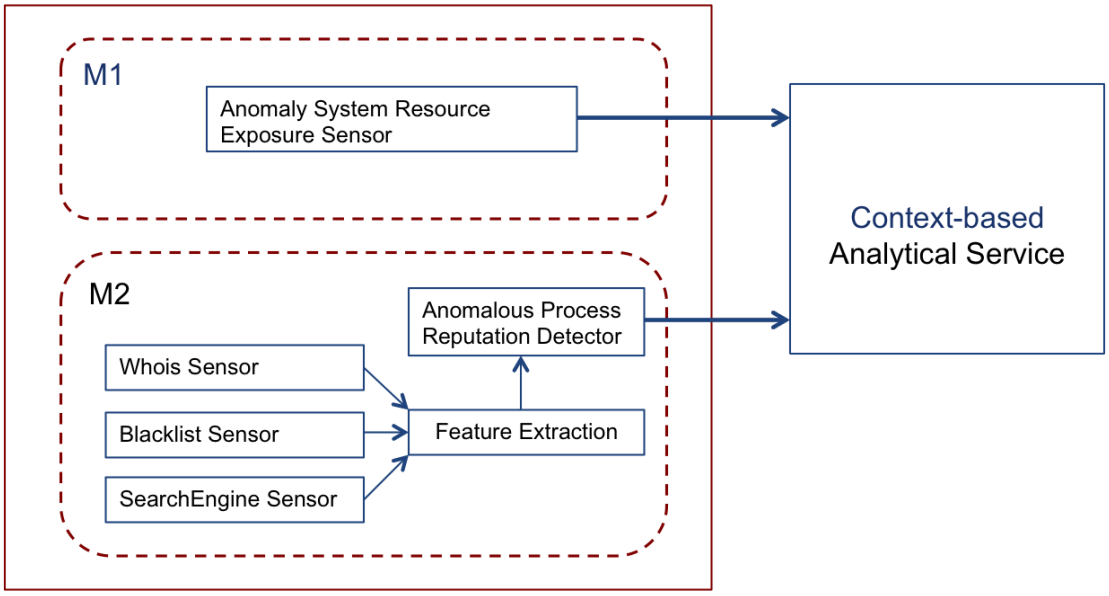


Figure 7.3: Process-Network purpose sensor architecture

gather online data on the domains contacted by each process and determines the reputation of the domains contacted by each process. The reputation of each process is therefore roughly determined from the reputation of the domains that it communicates with. It includes three sub-sensors:

- Whois Sensor: extracts top domain names used by processes for DNS queries and in the HTTP traces, from the received raw data, and collects domain registration information, such as domain creation time, domain expiration time, and number of domain registration servers. These domain registration features are critical in judging whether the domain is malicious or not since most malicious domains registered in the near time and has few domain registration servers.

- Blacklist Sensor: extracts domain names and IP addresses used for DNS queries and in the HTTP traces, from the received raw data, and determined whether a domain name or an IP address has been listed in a publicly distributed blacklist. The blacklist sensor uses the SpamHaus blacklist web service. This is a basic DNS translation service. For each IP address or domain name, the sensor appends .zen.spamhaus.org and performs a DNS IP translation. If a proper IP address is returned, then the given

host IP address or domain name is blacklisted. If no translation can be performed, then the host is not blacklisted.

- SearchEngine Sensor: apply the "xgoogle" google search engine to get another feature by leveraging Internet community-based knowledge. The sensor simulates a google search with the domain name as the searching keyword and parses the top 10 results returned from google. Then, checking whether the returned search result contains any malicious keywords such as "malicious", "botnet", "malware", "spam", "ddos", "identity theft", "IRC channel", "command and control", "C&C", "IRC server", "irc", "ircserver". If the matching value exceeds certain threshold, we assume that the domain is malicious.

Purpose Sensor analysis is conducted through examining the content of raw data received from the Purpose Sensors. The Content-based Analytics Service performs the following activities:

- Extracts domain names and IP addresses used for DNS queries and in the HTTP traces, from the received raw data, and determined whether a domain name or an IP address has been listed in a publicly distributed blacklist.
- Extracts top domain names used by processes for DNS queries and in the HTTP traces, from the received raw data, and collects domain registration information.
- Gathers information on the keywords returned by search engines for searches on the extracted domain names.
- Looks for Indicator of Compromise in the captured NetFlow traces in form of Flag Text. Flag text refers to a defining piece of text that is present in malicious network traffic (often a preamble) that ties it into a specific family or type of malware. For instance the Gh0st RAT family of malware will commonly use Gh0st as the flag text in its network traffic.
- Issues alerts based on results of above described analysis steps.

The purpose sensors using a set heuristics rules, examines the event content and might issue its own alerts that along with examined events are submitted into the system database for storage and further processing. If any alert is generated, the request for alert aggregation is issued to the Context-based Analytics Service.

7.4.2.3 Third-party Network Analysis Alert

Intrusion Detection Systems (IDS [66, 97]) are applications that monitor and assess network or system host activities for malicious, irregular, anomalies, and policy violations. Since detection is the main goal, these systems focus on identifying possible incidents and logging this information for further analysis in order to identify unauthorized or unapproved activity within a network. Network Intrusion Detection Systems (NIDS) and Host Intrusion Detection Systems (HIDS) provide the typical level of defense for computers and networks. For our system, we incorporate both of these two IDS: snort and OSSEC.

Snort [89] is a network based intrusion detection and prevention system. It performs real-time traffic analysis and alert reporting. Snort has rules to detect behaviors such as fingerprinting and port scans. These rules may be triggered as part of an APTs attack, but often generate too many false positives to be used effectively. Snort does not have any native paths to send data to our database, so we are using Barnyard to import data, which is an open source data interpreter built exclusively for Snort. .

Like Snort, OSSEC [70] is a scalable, multi-platform, real time open source IDS. The fact that it is cross-platform software that can be used throughout the network, allows network administrators to have a one-stop solution for implementing, collecting, and analyzing their networks from a host perspective. OSSEC has vast capabilities that include log analysis, file integrity checking, policy monitoring, rootkit detection, and real time active response. These characteristics complement Snort very well from the Host based perspective. OSSEC has its own built-in internal functions to send reporting data to the database.

Cisco's NetFlow [14] protocol is used to collect real-time IP network traffic as it enters or exits interfaces on each network node/hosts. The designed system builds a summary of distinguishable flow of communications between sources and destinations on a network

including source and destination IP addresses, source and destination ports, the protocol type, the type of service, interfaces on the router/firewall, if applicable. Multiple packets might be grouped or aggregated together under a common flow ID based on configurable timing threshold. The ID's context may also include some statistics such as number of packets covered by the flow, the total number of bytes, and duration and timing of those packets, The flow summary information is stored together with the packet payload content inside the database. Each parameter is provided to the APTs detection system Analytics Services to incorporate any abnormal traffic behaviors into the event correlation and APTs detection process.

7.4.3 Data Pre-processing and Normalization

Since the APTs detection system collects alerts from different sensors, and these alerts are encoded in different formats, we have to translate all attributes of each sensor alert into a common format. This translation requires that the syntax and semantics of a sensor alert are recognized. In order to express the causal dependency relationship between two collected system objects, such as processes, connections, files, etc., the alerts are normalized into the Source Object, the Sink Object, and the Action format as shown in Figure 7.4. This kind of object-oriented design is able to express relationships between alerts, which are an essential requirement of alert correlation. The detail information about run-time data collection and pre-processing is described in Appendix C.

7.4.4 Analysis for Traffic Anomaly Detection

Anomaly detection can be used to detect malicious or suspicious activities caused by intentionally or unintentionally induced attacks or defects in a network. Compared with signature-based intrusion detection systems (IDSs), the main advantage of anomaly detection techniques is their ability to spotlight previously unknown attacks. The majority of anomaly detection methods use some form of machine learning techniques to build a model of normal behavior from the observed normal traffic or activities, and then they measure the conformity

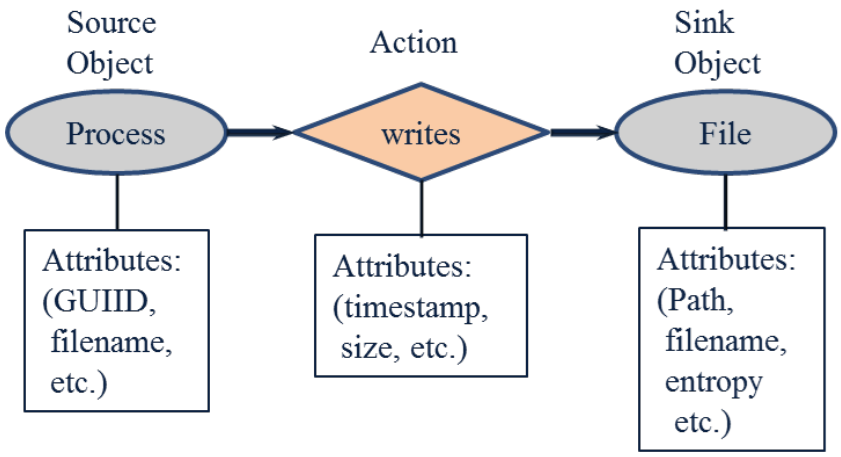


Figure 7.4: Alert normalization format

of new objects compared to the learned model of normality. Therefore, the effectiveness of anomaly detection crucially depends on the choice of features and careful analysis.

Various features have been studied in literature for general network intrusion detection, such as raw values of IP and TCP protocol headers, time and connection windows, and byte histograms. In this performance period, we particularly focus on determining network traffic features that can be feasibly extracted from industry standard network monitoring protocols such as NetFlow. By characterizing typical normal and abnormal traffic in an enterprise network, a set of reliable and sufficient traffic features have been identified. Using these features to build a baseline for each critical host, server, or subnet, the system can effectively detect the violations (i.e., abnormal traffic or activities) in a timely manner.

7.4.4.1 Characterization of Abnormal Traffic

Based on our initial study, most cyber attacks and system failures in an enterprise network will eventually generate some abnormal traffic (measured by volumes, durations, time of occurrences, and communication patterns, etc.), causing network protocol failures or error messages (e.g., HTTP, TCP, ICMP, DNS errors or failures), and/or execute a group of suspicious operating system (OS) level actions on a host in a short time period (e.g., adding an AutoRun key in the Registry, creating some .EXE files in the system directory, opening a number of TCP/UDP ports on the host, etc.). If the unique features and characteristics behind

the typical abnormal traffics can be identified, the accuracy and effectiveness of anomaly detection mechanisms can be improved significantly.

Traffic anomalies can be observed by processing traffic flows collected at the network level. The distributions of packet features (IP addresses and ports) observed in traffic flows can reveal both the presence and the structure of a wide range of anomalies. Figure 3 lists some unique features of traffic anomalies caused by typical attacks or system faults in an enterprise network. It shows that simple features can capture most anomalies/attacks. For instance, certain worms (e.g., NetBIOS) only propagate via specific ports (e.g., Port 139). Based on this observation, statistical models such as baseline analysis can be further developed to detect those traffic anomalies.

Type	Description	Features	Examples
DoS/DDoS	Prevent users from gaining desired resources	Spike in traffic flows and packet numbers to a host or service, typically last less than 20 minutes	A large number of packets sent to a single IP at ports that are frequent targets of DOS attacks (e.g., port 0)
Scan	Scanning a host or network for a vulnerable port/service or host	Spike in flows with similar number of packets from a dominant source, no dominant destination IP or port, typically last less than 10 minutes	NetBIOS scans for port 139, SQL Slammer scans port 1434 using a 376 byte long UDP packet, etc.
Worm	Self-propagate code exploits security flaws	Spike in traffic flows with a dominant destination port	MS SQL-Snake propagates through port 1433, Netbus Trojan propagates through port 12345, etc.
Alpha	Unusually high rate point to point byte transfer	Spike in traffic bytes or packet numbers in a dominant pair of hosts, typically in a short duration	Data exfiltration during off-work hours, etc.
Flash crowd	Unusually large demands for a resource/service	Spike in traffic flows to a dominant IP and port, typically short-lived	A large number of web requests to single IP (e.g., port 80)
System fault	Dramatic decrease in traffic exchanged between two IPs	Decrease in bytes, packets or flows, usually to zero, can last for long duration (hours) and in all instances	Downtime of a scheduled maintenance, measurement failure, etc.

Figure 7.5: Typical features of abnormal traffic

7.4.4.2 Feature Vectors for Traffic Anomaly Detection

Based on the traffic characteristics identified, a set of feature vectors can be defined to help security analysts effectively detect traffic anomalies and identify the corresponding threats in an enterprise network. In our work, a feature vector is defined as a vector that

contains information to describe an objects important characteristics. The raw data we first considered is the Cisco NetFlow data, which consists of flow records that have been exported by NetFlow enabled routers or network monitors. We transform NetFlow records into datasets with a small number of features for predefined time intervals and service-specific port numbers. Our goal here is to identify a set of sufficient features and use this feature vector to build a baseline for each critical host, server, service, or subnet in a network, based on the observed historical data. Using this baseline as the prior knowledge, abnormal traffic can be quickly detected when any violation has to be observed. To further reduce the false positive, this anomaly detection result is correlated with detection results of other sensors (e.g., Snort, OSSEC, and purpose sensors). The validated detection result is aggregated, prioritized, and notated to provide users an accurate, clear and deep understanding of cyber situational awareness in their network.

We first define a set of feature vectors to build a baseline for each critical host and subnet in a target enterprise network. For the granularity, we start with one hour for each day of a week. The historical data used to build a baseline is one week. We merge/aggregate the back log with another set of newly accumulated data on a weekly basis. Essentially, for the egress traffic collected at the edge routers or firewalls, each host (identified by a source IP address in a NetFlow record), in each hour in each day of a week, we calculate:

- Number of unique connections.
- Duration (or connection time) of each connection.
- Distribution of connection duration.
- Total traffic volume (e.g., number of packets or bytes) for each connection.
- Average traffic volume (e.g., number of packets or bytes) for each connection.
- Distribution across connections.
- Protocol/port distribution (e.g., TCP, UDP, ICMP) across connections in traffic volumes.

- Protocol/port distribution across connections in number of connections.

At the network level, we also need to build a baseline for network level traffic anomaly detection, based on the statistics of the traffic flows and connections generated by critical hosts in a network. Essentially, in order to model the network behaviors at different time (e.g., each hour of each day in a week, including the business hours, after hours, holidays, weekends, etc.), we need to consider the following feature for the entire network or each subnet in a network:

- List of top Source IP addresses that generate the most traffic volumes in bytes or packet numbers.
- List of top Destination IP addresses that generate the most traffic volumes in bytes or packet numbers.
- List of top conversations (SIP/Sport-DIP/Dport pairs) that generate the most traffic volumes in bytes or packet numbers.
- List of top protocols that generate the most traffic volumes in bytes or packet numbers.
- The overall total amount of bytes generated by a subnet or network.
- The overall total amount of packets generated by a subnet or network.

7.4.4.3 Baseline Analysis for Traffic Anomaly Detection

Baseline network traffic is watching network traffic and figuring out what is normal in everyday traffic, and knowing when the traffic pattern of any internal host changes. By setting a network traffic baseline, network administrators can define what is normal for enterprise networks and identify patterns that indicate anomalies. In enterprise networks, each internal host is either a consumer or producer of traffic. They are known to normally download files from a shared drive on a typical day, upload files, or a mixture of both. Security analysts need to know when the host changes its regular traffic patterns. For example, when

a host is normally a consumer of data and becomes a producer of data, this could be an indication that it may be an infected host that is starting to exfiltrate data.

In the simplest terms, a network traffic and performance baseline is a set of metrics used in network traffic monitoring to define the normal working conditions of an enterprise network infrastructure. These baselines can be used by security analysts for comparison to catch changes in traffic that could indicate a problem. In this effort, we focus on identifying the time intervals that show anomalous traffic behavior that could be caused by network malfunctions or malicious attacks.

The processing steps of our approach can be summarized as training data containing flow records of both normal and anomalous traffic that are transformed into feature datasets, then the datasets are divided into different clusters for normal and anomalous traffic using advanced clustering algorithm, and the resulting cluster centroids are deployed for fast detection of anomalies in new monitoring data based on simple distance calculations.

Algorithm 2 shows the baseline analysis algorithm for traffic anomaly detection. Using the identified baseline of each individual host (or subnet) and the real-time traffic patterns as the inputs, we can calculate the suspicious level for each critical host. If the accumulated suspicious level exceeds the pre-defined threshold, an alarm will be raised and reported to the security analysts. Correlating this analysis result with detection results generated by other sensors, such as Snort, OSSEC, and purpose sensors, the detection rate and accuracy could be significantly improved.

7.4.5 Cloud-based Data Storage and Management

The APT detection storage utilizes the cloud-based Accumulo database technology that uses indexing, normalization, and tagging for effective data storage, querying, and analysis. Apaches Accumulo is an open-source, distributed, column-oriented store modeled after Googles Bigtable. Accumulo provides random, real-time, read/write access to large datasets atop clusters of commodity hardware. Accumulo leverages Apache Hadoop Distributed File System, the open source implementation of the Google File System. In addition to Google

Algorithm 2: Baseline Analysis Algorithm for Traffic Anomaly Detection

Input:
- Baseline of Host (or Subnet) i:

$$BaseLine(Host_i) = [b_1, b_2, ..., b_n]$$

- Current behavior of Host (or Subnet) i:

$$CurrentBehavior(Host_i) = [f_1, f_2, ..., f_n]$$

Output:
- Suspicious Level of Host i from 0 to N : 0 is normal, N stands for extremely abnormal.

Result:
- For each feature/property, compare its baseline value and current value
- index++, if there is a violation on property i
- Suspicious Level = index
- if (Suspicious Level >Threshold), then raise an alarm (e.g., "Host i's behavior is abnormal, the suspicious level is X!")

Bigtable features, Accumulo features automatic load balancing and partitioning, data compression, and fine grained security labels.

Accumulo is a sorted, distributed key/value store, which provides robust, scalable, high performance data storage and retrieval system. Each key is composed of (Row ID, Column, Timestamp) as shown in Table 7.1. Rows in the table are stored in contiguous ranges (sorted by key) called tablets. Tablets are managed by tablet servers, with a tablet server running on each node in a cluster. Accumulo provides a server side Iterator model that helps increase performance by conducting large computing tasks directly on the servers and not on the client machine, thus avoiding the need to send large amounts of data across the network. Accumulo also provides a Batch Scanner client API that allows it to reduce multiple range scans into one merged client request. The Batch Scanner is used to process a large number of range scans quickly, by merging all the scans into the smallest number of scans possible to return the correct data set.

Table 7.1: Basic Accumulo table format

Key					Value
Row ID	Column			TimeStamp	
	Family	Qualifier	Visibility		
...	...	...			...

Table 7.2: APT detection system Accumulo table example

Key					Value
Row ID	Column			TimeStamp	
	Family	Qualifier	Visibility		
EntityID1	Source:Process	ProcessGuid:1029		Time1	
EntityID1	Source:Process	ProcessId:12		Time1	
EntityID1	Source:Process	User:IAI02		Time1	
EntityID1	Source:Process	CurrentDirectory:”C:”		Time1	
EntityID2	Sink:Endpoint	L3Protocol:IP		Time2	
EntityID2	Sink:Endpoint	L4Protocol:udp		Time2	
EntityID2	Sink:Endpoint	IP:10.7.200.1		Time2	
EntityID2	Sink:Endpoint	Port: 80		Time2	
EntityID2	Sink:Endpoint	IsIPv6:No		Time2	
EntityID2	Sink:Endpoint	Interface:”eth0”		Time2	
...	...	...		...	
...	...	...		...	

Alert data are originated from diverse 3rd party IDS/HIDS systems. The Data Preprocessing Service extracts standard (for each alert reporter) fields. Accumulo tables can support any number of columns without specifying them beforehand. In order to support lookups via more than one attribute of an entity, additional indexes can be built. Raw Alerts are also stored in the database for history analysis and the training purpose. Both raw data and conditioned (normalized) data are stored together, whereas raw data is represented as an additional column containing ASCII text. The system also supports alert aggregation, correlation and annotation. It creates custom built Iterators for alert aggregation, and utilizes specialized Combiners and MapReduce jobs for alert correlation and annotations. The example database table design is shown in Table 7.2.

7.4.6 Context-based Threat Analytic

When a new alert is reported by the purpose sensor, the Threat Analytic Service is triggered and will query the Database to retrieve relevant events that have similar "Source" or "Sink" Objects within certain time threshold, and creates an event relationships in the form of Dependency and Causality (D&C) graphs by connecting with the same Source Object or Sink Object. D&C graphs are collections of system-level causal events that connect the collected system objects, like processes, connections, files, etc as shown in Figure 7.6. The events chains form the links that allow for deducing timelines and context relationships of events connecting various objects, since each event action has a time threshold associated with it, which is the maximum time that an event can occur and be considered relevant for that pair of objects. The starting or detection point for the linking process is an alert generated by either the third-party IDS/HIDS software or by the Purpose Sensors. The term "detection point" here refers to the state on the local computer system that alerts the administrator to the potential intrusion or compromise. For example, a detection point could be a deleted, modified, or additional file, or it could be a process that is behaving in an unusual or suspicious manner. Time based Backward Event Tracking: performing backward event chain tracking for each event chain within the specified historic time period, starting from the specified time. As shown in Figure 7.7 and Figure 7.8, we correlate the corresponding objects into a "wide net" step by step that reflect the context events of the triggering alert.

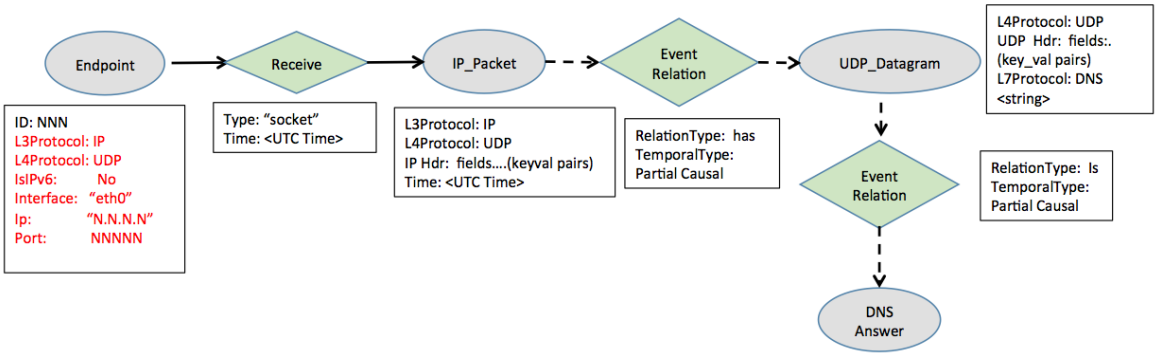


Figure 7.6: Dependency and Causality graph event chaining example 1

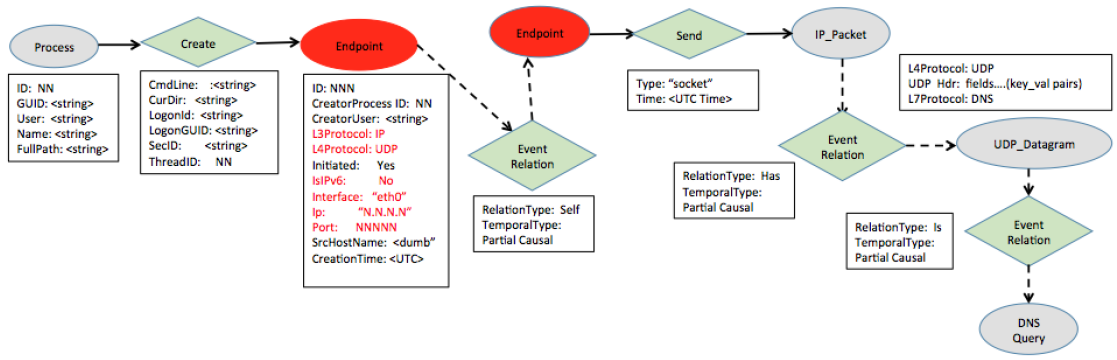


Figure 7.7: Dependency and Causality graph event chaining example 2

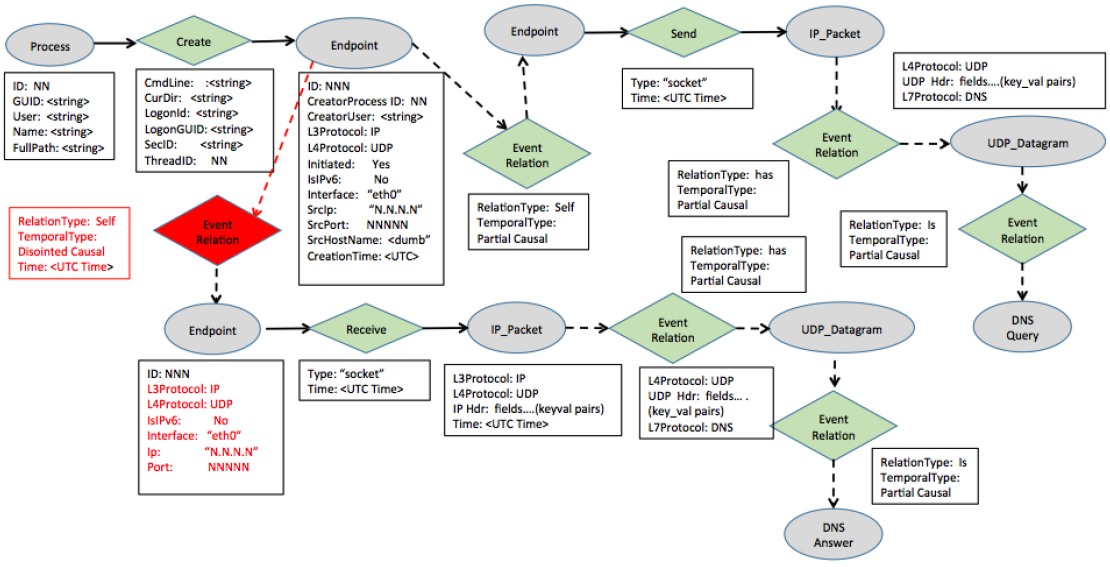


Figure 7.8: Dependency and Causality graph event chaining example 3

APT narratives are viewed as graphs of behavior primitives that capture aggregate description of threat agents. These threat narratives can represent characteristics that are shared between a group of hosts and across the target network. We represent each threat as a set of events (behavior primitives) with corresponding extracted features. The events that represent threat behavior primitives are chained together to form dependency and causality (D&C) graph. Each threat, hence, is defined as a D&C graph or an event chain as shown in Figure 7.6. Each Object and Action in the graph is assigned a set of attributes to represent "extractable" feature sets. We use APT Profiles to perform assessment a probability of APT

presence on the target network.

APT Profiles comprise the Actions that describe the source-to-sink object relationship effect as either affecting or controlling. For example, changing the contents of a file or creating a child process are examples of controlling effect. In context of APT an attacker installs a backdoor by modifying an executable file, then creates a process that executes it. Some examples of affecting effects are changing a file's access time, creating a filename in a directory, or enumerating a registry key. Typical APT pattern combines the event with affecting and with controlling action types. Using APT patterns with a higher percentage of controlling events for probability of occurrences computation against constructed event chains would result in a higher similarity score.

APTs Profiles is also represented as a D&C graph of behavior primitives that capture aggregate description of threat agents. These threat narratives can represent characteristics that move between a group of hosts and across the target network. APTs Profiles comprise the actions that describe the source-to-sink object relationship effect as either affecting or controlling. For example, changing the contents of a file or creating a child process are examples of controlling effect. In context of APT, an attacker installs a backdoor by modifying an executable file, and then creates a process that executes it. Some examples of affecting effects are changing a file's access time, creating a filename in a directory, or enumerating a registry key. Typical APTs pattern has a combination of the event with affecting and with controlling action types. Figure 7.9 show the example D&C graph.

We follow object-oriented paradigm for Events, Objects and Actions. This allows extending the set of original APT Profiles by deriving new Objects and Actions from existing templates. Using parent-child object relationship allows applying generalization principle for object comparison during D&C graph traversal and inexact event matching. The system provides an option for a user to create custom APT Profiles. A new APT profile is derived from existing APT profile templates.

The Context-based Analytics Service performs assessment of a probability of APTs presence on the target network. The probability assessment is implemented by iterating through the list of constructed D&C graph/event chains (built using one or more aggregated

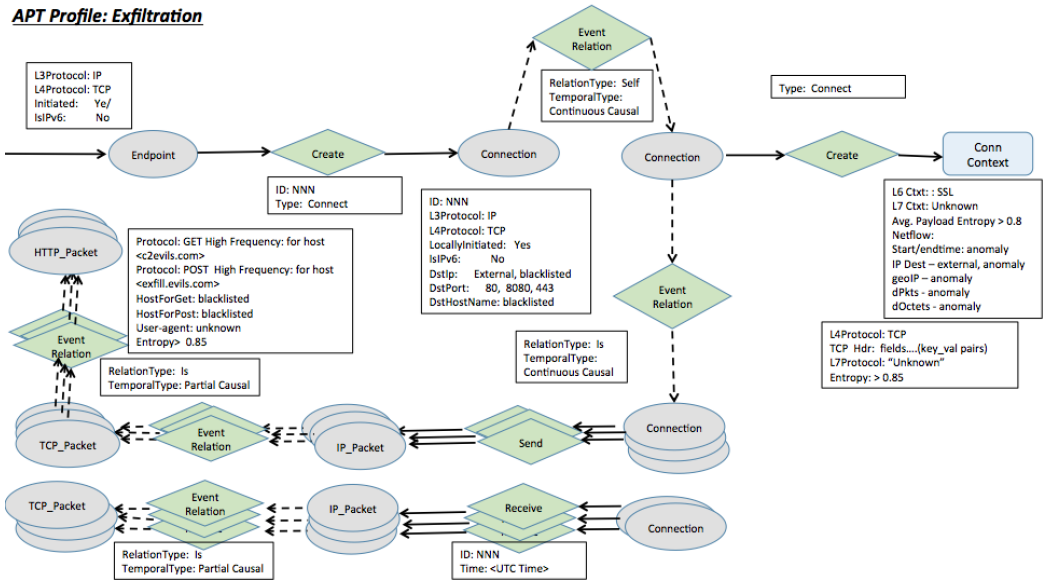


Figure 7.9: APT Profile for exfiltration attack

alerts as a starting point), traversing of each graph, and performing inexact matching and computing distance metrics between a generated alert D&C graphs and an available APTs profile. The aggregate similarity score is calculated for each combination of an event chain and an APTs profile. Both the Object Attributes and Action Attributes for each Event are utilized to form feature vector sets used for Jaccard similarity distance calculations. The Jaccard distance measures subset similarity as the ratio of the number of feature elements in the intersection and the union as shown in the following formula.

$$d_j(A, B) = 1 - \frac{|A \bigcap B|}{|A \bigcup B|} \tag{7.1}$$

7.5 Evaluation and Results

In order to perform functional testing, obtain realistic results, and evaluate the performance of the designed system, we leveraged the hardware and cloud based testbed platform as shown in Figure 7.11. It consists of APTs cluster, Intranet Service cluster, DMZ server, Workstation cluster, and the analysis cluster. The APTs cluster is used to model various APTs scenarios based on discovered attack vectors and known adversarial attacks. We have

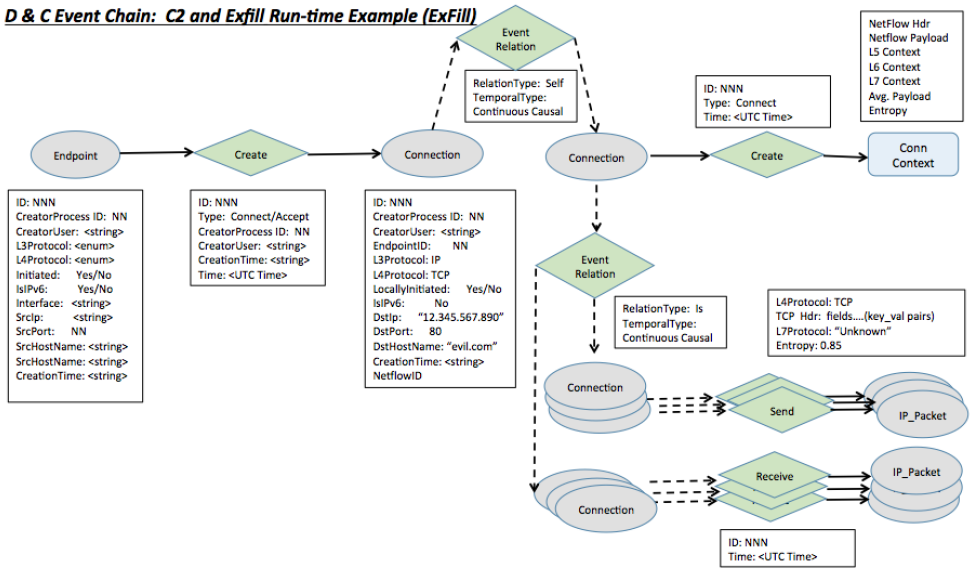


Figure 7.10: Threat Dependency and Causality graph exfiltration run-time pattern

simulated three representative APTs attack to consider are a slow port scan attack, a brute force password attack, and data exfiltration attack.

7.5.1 Slow Port Scan Attack

The port scan [102] is the most popular way for someone to conduct reconnaissance of a network, making its detection critical. While a very intrusive and quick scan can be easy to detect, a common APTs tactic is to slow down the rate of the port scan below the target network's detection threshold since time is not a critical resource to the attacker. This behavior cannot be easily detected since an individual port scan packet by itself is not necessarily malicious and may instead be simply a mistaken attempt to connect to a non-existent service. An IDS uses a threshold to determine whether to generate an alert, forcing operators to choose between either receiving no alerts for an APT's port scan or receiving a huge volume of alerts, nearly all of which are false positives. To mimic the behavior of a realistic APTs attack, we used NMAP to port scan the target host at a slow rate to avoid detection by the IDS. For testing purposes, we used a max rate of one packet every 10 seconds. This was detected and reported by Snort via custom and standard rules. The data provided by Snort was collected and stored in database.

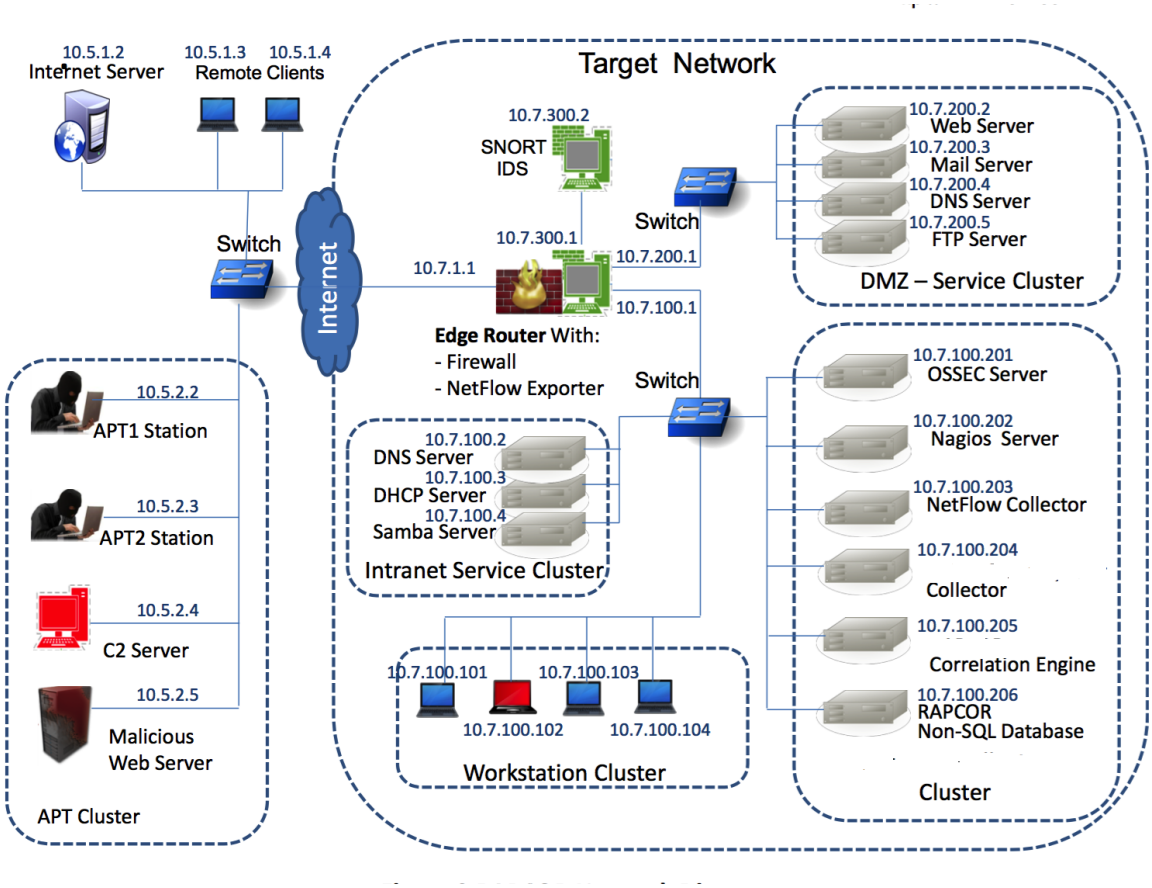


Figure 7.11: APTs detection testbed connection diagram

In order to detect the slow port scan attack, APTs detection system takes the data provided by Snort related to port scans. Under normal conditions, this Snort data has a very high false positive rate and so it is typically ignored or disabled. To solve this problem, APTs detection system combines received Snort data with our human input sensor data. The data is combined by looking at every alert from Snort and then checking to see if the user was active during the 6 seconds before and 3 seconds after (we are doing this to account for a timestamp error). Once APTs detection system correlates the data together, it is straightforward to see if the user triggered the process that requested each connection. If not, then APTs detection system annotates the Snort data as an actual port scan. The correlated event is saved and passed to the investigation engine to be grouped and prioritized before being displayed to the user.

Table 7.3 shows the result of a slow port scan attack. In total there were 328,665 data points collected from Snort containing two actual slow port scan attacks. As shown in the table, both slow port scan attacks were detected. The 2 port scan attacks were found to be malicious based on our detection approach described earlier in this section. Due to the long duration of a slow port scan, there were a lot of other legitimate noise packets sent out and captured within the dataset which is why there are a lot of data points for this attack scenario.

Table 7.3: Slow port scan result

Metric	Value
Total Slow Port Scans	2
Total Detected Slow Port Scans	2
Total Data Points	328,665

7.5.2 Bruce Force Password Attack

The second attack that we used for evaluation is the brute force password attack [49, 78]. In a brute force password attack, an attacker guesses many passwords quickly using a dictionary of possible passwords. As with a port scan attack, in a standard brute force attack, there are very many incorrect password attacks within a short period of time, making it easy for an IDS to employ threshold-based detection. An APTs will instead slow down the guess rate to the point that it is indistinguishable from the rate at which users accidentally enter invalid passwords due to simply pressing the wrong key. OSSEC, our host-based IDS, can create an alert for each individual attempt at a failed login. Such an alert is not typically generated due to the very high false positive rate that would result.

For our simulation, we use remote desktop to simulate logging into a machine via a password. We manually input the wrong password to create failed login attempts. Since failed logins are a very common occurrence within a network, most of the alerts received from OSSEC are false positives. To eliminate the false positives, APTs detection system correlates the raw OSSEC alerts with the human input sensor to see if the computer was actively being used by a human. APTs detection system can then reliably determine which of the many OSSEC alerts are false positives.

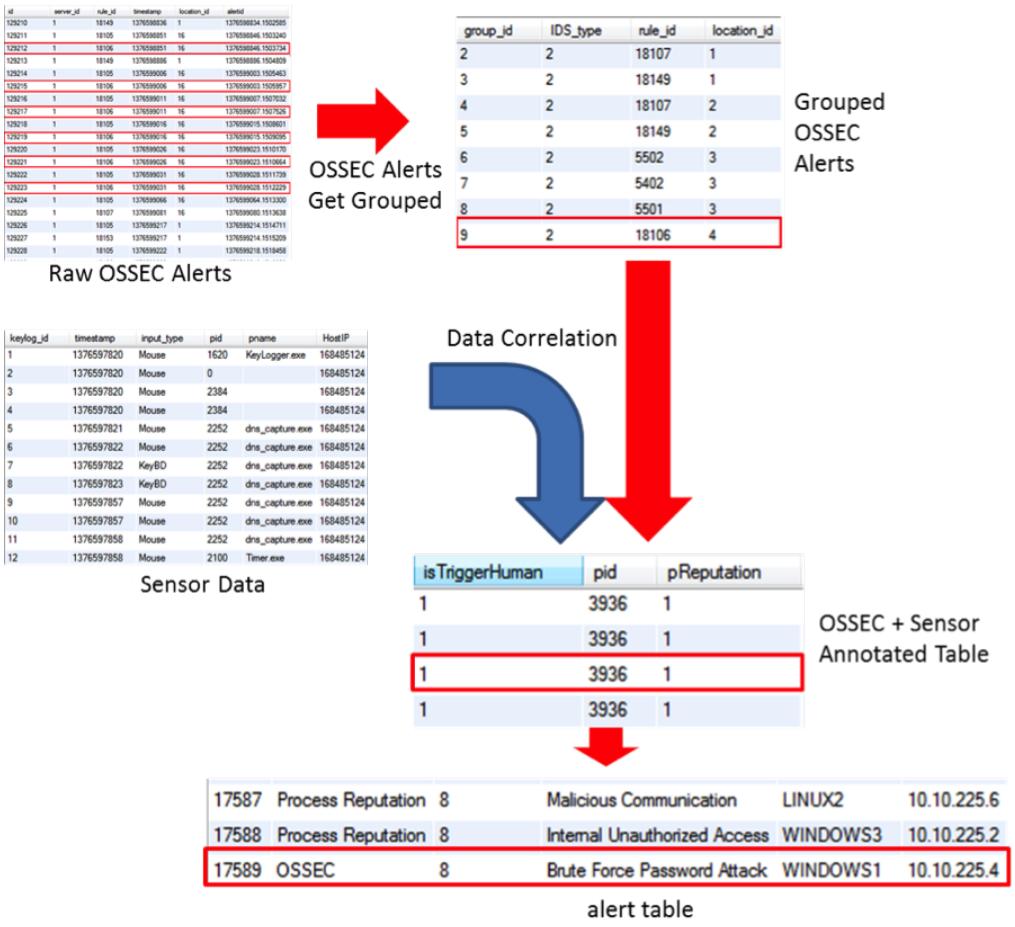


Figure 7.12: Brute force attack detection

Figure 7.12 shows the detection of a brute force attack. The raw OSSEC alerts are grouped by rule id. These grouped OSSEC alerts are then correlated with the purpose sensor data to create the OSSEC and Sensor annotated table. This table information becomes the Alert History data used for subsequent processing. Based on the input from the human purpose sensor and process reputation sensor APTs detection system determines whether received alerts from OSSEC are malicious. The established context provides the data that allows an operator to look closer at the potentially malicious event and determine whether a brute force attack is occurring and whether further action is required.

Table 7.5 shows the overall result summary of executed test scenarios. The total data points within the considered data set is 352,993 points. Out of a total of 240 sign on errors

40 were actual malicious errors and all of them were correctly detected by APTs detection system with no false positives. Furthermore, all 40 malicious login errors were correctly grouped into four brute force attacks.

Table 7.4: Bruce force password attack result

Metric	Value
Total Sign On Errors	240
Total Malicious Login Errors	40
Total Grouped Brute Force Attacks	4
Total Data Points	352,993

7.5.3 Data Exfiltration Attack

Many APTs attacks are designed with a goal of getting information from within the network back to the attacker. Based on this observation, the third type of APTs behavior we considered is a data exfiltration attack [58, 7]. When an attacker tries to exfiltrate data out of a network, it is very unwise to send many files out. The attacker wants to be stealthy and leave as small of a footprint as possible, especially while they are still in the system. A common way to exfiltrate data is to compress the data into a common file type, possibly encrypt the data to evade higher levels of detection, and then send it out of the network. To conduct a simulation of this attack, we wrote python scripts to compress, send, and receive a data file using the RAR file compression format. We then created a Snort rule to detect any outbound RAR files. This rule by itself has a high very false positive rate, as any legitimate RAR file, not only those sent by an APT, will trigger the rule.

Figure 7.13 shows how APTs detection system detects a data exfiltration attack. A Snort rule is used to detect .rar files. These Snort events are then correlated with APTs detection system's process reputation sensor and the human input data to determine whether the data exfiltration was malicious or not. If the data exfiltration is not performed by a human action but instead by an automated process with low reputation score, APTs detection system annotates the event accordingly.

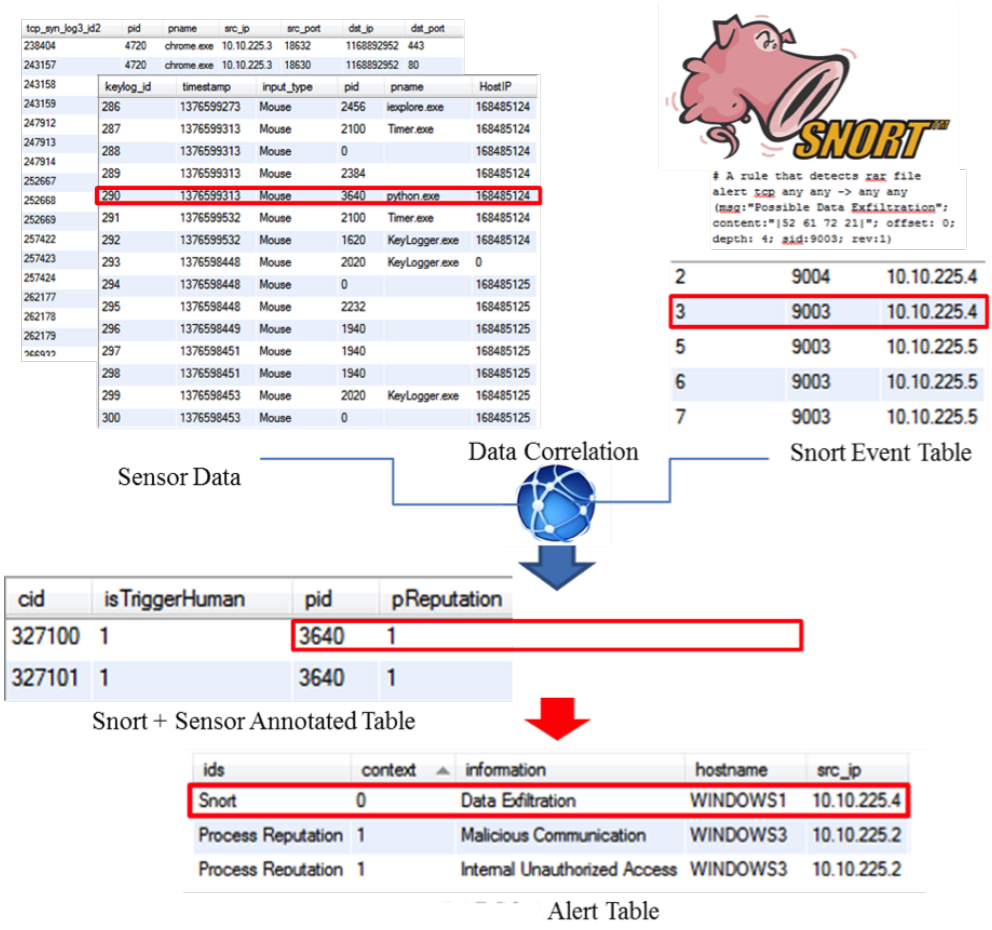


Figure 7.13: Data exfiltration attack detection

Table 7.5 shows the overall result summary of executed test scenarios. Out of a total of 6 exfiltration alerts within the considered data set only two were actually malicious exfiltration attacks. the system was able to successfully detect these two malicious exfiltration attacks. The total data points within the considered data set was 529,235 points, thus demonstrating clear effectiveness of the designed system to reduce the number of false positives.

Table 7.5: Data exfiltration attack result

Metric	Value
Total Exfiltration Alerts	6
Total Malicious Exfiltration Attacks	2
Total Detected Malicious Alerts	2
Total Data Points	529,235

7.5.4 Evaluation Result Discussion

When we first defined the requirements for the APT detection system, we focused on attacks that are very common steps for APTs. Since APTs are very sophisticated and require many working parts to pull off such an attack, we set our sights on demonstrating the detection of the parts of the APT attacks that are both very important to the attackers mission and difficult for current techniques to detect without generating many false positives. We observe that many APT behaviors are detectable in some way, however due to the high false positive rate these detectors are turned off or set to thresholds that do not catch an APTs actions.

From our tests, we demonstrated the feasibility of the detection system. First, the system is able to successfully discern whether OSSEC failed login triggers are malicious or are due to a user signing onto a computer incorrectly. The purpose sensors give the ability to know when a false positive has occurred, allowing this data to be removed from the final output provided to the user. We have also included Snort IDS triggers into the system to detect malicious actions including port scans of various scopes and intensities. Finally, our system leverages high false positive Snort rules to detect any possible data leaving the network without human interaction. The current results show that we can detect three steps

in an APT attack process, 1) understanding the network, 2) getting access to a machine, and 3) exfiltrating data.

Each of the behaviors chosen are representative of larger classes of attacks, and the system can similarly reduce the false positive rates for related attacks. It is important to note that reducing the false positive rate does not only reduce operator burden. In our results, we have shown that the systems ability to take high false positive data as input and filter out only the true positives makes many sources of raw data which have been previously unusable now become reliable detectors of APTs. This change allows a corresponding decrease in the false negative rate, meaning that more APTs will be detected.

7.6 Conclusion

We have developed a novel non-signature based system for Advanced Persistent Threat (APT) detection. By parsing the raw alerts from Purpose Sensors and third-party network analysis tools into Source Object, Action, and Sink Object format, we can create a Dependency and Causality graph containing all relevant events within a certain time threshold. Through calculating the Jaccard distance between the alert Dependency and Causality graph and APTs profiles, we can evaluate whether the suspicious alert is a real APTs attack or a false alarm. Through simulation in a cloud-based testbed, we demonstrated that the designed system could effectively detect important types of APTs behaviors such as slow port scan attacks, slow brute force attacks, and data exfiltration attacks, as well as significantly reduce the amount of false positives that are reported to the cyber analysts.

Chapter 8

FUTURE WORK

In the future, we are going to implement dynamic assessment techniques to evaluate trainees' performance on CSA. We also plan to use the security training lesson plans in undergraduate cyber security courses to validate the effectiveness of cyber security training systems and design more effective training scenarios based on trainees' reflection. Furthermore, how to evaluate team CSA performance remains a critical issue.

For the APT detection part, we are going to expand the current APT detection testbed to include hundreds or thousands of diverse network machines to create a realistic enterprise level test environment and emulate more complex APT attacks to validate the proposed system. Such a testbed will also significantly contribute to analysis and research of new advanced APT attacks and also enable us to develop new components that will further advance the capabilities of APT detection such as using forward tracking analysis for impact assessment.

We may integrate more advanced mission data and contextual information. Leveraging more advanced mission information will improve the functionality of the APT detection system. We have already developed extensive experience in this area of research during our work on the CREACT (Advanced Network Security Metrics for Cyber Resilience and Asset Criticality Measurement in Mission Success) project. The CREACT project has considered the problem of efficiently evaluating the networks health and ability to achieve the overall mission success given that available network assets support different missions with different priorities. This background will contribute advanced, value-based goal models and efficient mission-to-asset mapping, resource allocation, vulnerability assessment, threat analysis and impact mitigation techniques for cyber resilience and asset criticality modeling, evaluation and measurement. These capabilities will significantly enhance the current mission related

context input and will enable the detection system to effectively evaluate, prioritize, identify and allocate critical cyber assets to ensure mission success and provide more focused detection against sophisticated attacks.

The current correlation engine provides an effective means of annotating received alerts with purpose sensor information to create meaningful event contexts which enable the cyber security operator to quickly identify and focus efforts on the most critical attacks. Our future efforts will focus on enhancing the existing rules by integrating additional information obtained from new network and purpose sensors, advanced mission data, and by allowing an operator's and/or cyber analyst's feedback to customize the system's presentation and handling of data to best meet their needs.

We are also going to build real-time network status visualization GUI for the APT detection system. The visualization engine periodically (using configurable time intervals) checks the database for the outputs/updates of the corresponding analytic services, and then displays/updates the events on the topology and swim-lane views. Cyber analysts can receive a prioritized set of new alerts; query the history of cyber threats with diverse attributes (e.g., time interval, priority, annotation, etc.) to gain additional contextual knowledge.

BIBLIOGRAPHY

[1] Mansour Alsaleh and P. C. Oorschot. Network Scan Detection with LQS: a Lightweight, Quick and Stateful Algorithm. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 102–113, Hong Kong, China, 2011. ACM.

[2] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, Graph-based Network Vulnerability Analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 217–224, New York, NY, USA, 2002.

[3] Michael Lyle Artz. NetSPA : a Network Security Planning Architecture. Master's thesis, Massachusetts Institute of Technology, Boston, MA, USA, 2002.

[4] Thomas Augustine and Ronald C. Dodge. Cyber Defense Exercise Meeting Learning Objectives thru Competition. In *In Proceedings of the 10th Colloquium for Information Systems Security Education*, 2006.

[5] Fatemeh Azmandian, Micha Moffie, Malak Alshawabkeh, Jennifer Dy, Javed Aslam, and David Kaeli. Virtual machine monitor-based lightweight intrusion detection. *SIGOPS Oper. Syst. Rev.*, 45(2):38–53, July 2011.

[6] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[7] Elisa Bertino and Gabriel Ghinita. Towards mechanisms for detection and prevention of data exfiltration by insiders: Keynote talk paper. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 10–19, New York, NY, USA, 2011. ACM.

[8] Monowar H Bhuyan, D.K Bhattacharyya, and J.K. Kalita. Surveying port scans and their detection methodologies. *Comput. J.*, 54(10):1565–1581, October 2011.

[9] Fatih Emre Boran, Serkan Genc, Mustafa Kurt, and Diyar Akay. A multi-criteria intuitionistic fuzzy group decision making for supplier selection with topsis method. *Expert Syst. Appl.*, 36(8):11363–11368, October 2009.

[10] Stephen R. Byers and Shanchieh J. Yang. Real-time fusion and projection of network intrusion activity. In *Information Fusion, 2008 11th International Conference on*, pages 1–8. IEEE, 2008.

[11] Sergio Caltagirone, Paul Ortman, Sean Melton, David Manz, Kyle King, and Paul Oman. A Rapidly Reconfigurable Computer Lab for Software Engineering Security Experiments and Exercises. In *19th Conference on Software Engineering Education and Training Workshops, CSEETW '06*, 2006.

[12] Michael A. Champion, Prashanth Rajivan, Nancy J. Cooke, and Shree Jariwala. Team-Based Cyber Defense Analysis. In *2012 IEEE International Multi-Disciplinary Conference onCognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2012.

[13] Joel Chinnow, Rainer Bye, Stephan Schmidt, Karsten Bsufka, Seyit Ahmet Camtepe, and Sahin Albayrak. An Extensible Simulation Framework for Critical Infrastructure Security. In *AI Laboratory, School of Electrical Engineering and Computer Science of the Berlin Institute of Technology, Technical Report: TUB-DAI 09/09-1*, 2009.

[14] Cisco. *NetFlow Analyzer User Guide - ManageEngine*.

[15] Eric Cole. *Advanced Persistent Threat: Understanding the Danger and How to Protect Your Organization*. Syngress Publishing, 1st edition, 2013.

[16] Nancy J. Cooke, Michael Champion, Prashanth Rajivan, and Shree Jariwala. Cyber situation awareness and teamwork. In *EAI Endorsed Transactions on Security and Safety*, 2013.

[17] Nancy J. Cooke, Anita DAmico, Mica R. Endsley, Emilie Roth, and Eduardo Salas. Perspectives on the Role of Cognition in Cyber Security. In *in Proceedings of the Human Factors and Ergnomics Society 56th Annual Meeting*, 2011.

[18] Weidong Cui, R.H. Katz, and Wai tian Tan. Binder: An extrusion-based break-in detector for personal computers. In *Computer Security Applications Conference, 21st Annual*, 2005.

[19] Oliver Dain and Robert K. Cunningham. Fusing a heterogeneous alert stream into scenarios. In *In Proceedings of the 2001 ACM workshop on Data Mining for Security Applications*, pages 1–13, 2001.

[20] Anita D'Amicoa, Kirsten Whitleyb, Daniel Tesonea, Brianne O'Briena, and Emilie Rothc. Acheiving cyber defense situational awarness: A cognitive task analysis of information assurance analysts acm. In *In Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, 2005.

[21] Anita D'Amicoa, Kirsten Whitleyb, Daniel Tesonea, Brianne O'Briena, and Emilie Rothc. Achieving cyber defense situational awareness: A cognitive task analysis of information assurance analysts. In *In Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2005.

[22] Irfan Delia and Naim agmanb. Intuitionistic fuzzy parameterized soft set theory and its decision making. In *in Applied Soft Computing*, pages 109–113, 2015.

[23] Brian Duffy. Network Defense Training through CyberOps Network Simulations. In *In Proceedings of the Modeling, Simulation, and Gaming Student Capstone Conference*, 2008.

[24] Chris Eagle and J. L. Clark. Capture-the-Flag: Learning Computer Security Under Fire. In *In Proceedings of the Sixth Workshop on Education in Computer Security (WECS)*, 2004.

[25] M. R. Endsley. Toward a theory of situation awareness in dynamic systems. In *in Human Factors*, 1995.

[26] L. Feinstein and D. Schnackenberg. Statistical Approaches to DDoS Attack Detection and Response. In *In Proceedings of the DARPA Information Survivability Conference and Expostion(DISCEX)*, 2003.

[27] Ulrik Franke and Joel Brynielsson. *Cyber situational awareness e A systematic review of the literature*. Computers & Security, 2014.

[28] 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*, QoP '08, pages 23–30, New York, NY, USA, 2008.

[29] Paul Giura and Wei Wang. A context-based detection framework for advanced persistent threats. In *Proceedings of the 2012 International Conference on Cyber Security*, CYBERSECURITY '12, pages 69–74, Washington, DC, USA, 2012. IEEE Computer Society.

[30] R. J. Guild. Design and Analysis of a Model Reconfigurable Cyber-Exercise Laboratory (RCEL) for Information Assurance Education. Master's thesis, Naval Postgraduate School, Monterrey, California, USA, 2004.

[31] Ramakrishna Gummadi, Hari Balakrishnan, Petros Maniatis, and Sylvia Ratnasamy. Not-a-bot: Improving service availability in the face of botnet attacks. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, pages 307–320, Berkeley, CA, USA, 2009. USENIX Association.

[32] Ali Hamieh and Jalel Ben-Othman. Detection of jamming attacks in wireless ad hoc networks using error distribution. In *Proceedings of the 2009 IEEE International Conference on Communications*, ICC'09, pages 4831–4836, Piscataway, NJ, USA, 2009.

[33] Andrew P. Hansen. Cyber Flag A Realistic Cyberspace Training Construct. Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, USA, 2008.

[34] Paul Harrison. IA Education, Training and Awareness Online Training Catalog of the Information Assurance Support Environment (IASE) of the US DoD Defense Information Systems Agency (DISA). http://iase.disa.mil/eta/online-catalog.html.

[35] Paul Harrison. Robust topological sorting and tarjan's algorithm in python. http://www.logarithmic.net/pfh/blog/01208083168/.

[36] F. Herrera, E. Herrera-Viedma, and J. L. Verdegay. A model of consensus in group decision making under linguistic assessments. *Fuzzy Sets Syst.*, 78(1):73–87, February 1996.

[37] John Homer, Ashok Varikuti, Xinming Ou, and Miles A. Mcqueen. Improving attack graph visualization through data reduction and attack grouping. In *Proceedings of the 5th International Workshop on Visualization for Computer Security*, VizSec '08, pages 68–79, Berlin, Heidelberg, 2008.

[38] John Homer, Su Zhang, Xinming Ou, David Schmidt, Yanhui Du, Raj S. Rajagopalan, and Anoop Singhal. Aggregating Vulnerability Metrics in Enterprise Networks Using Attack Graphs. *J. Comput. Secur.*, 21(4):561–597, July 2013.

[39] Zequn Huang and Chien-Chung Shen. Context-aware alert correlation for advanced persistent threat detection. In *the Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec 2015)*, 2015.

[40] Zequn Huang, Chien-Chung Shen, Sheetal Doshi, Nimmi Thomas, and Ha Duong. Cognitive task analysis based lesson plans for cyber situation awareness training and assessment. In *9th World Conference on Information Security Education (WISE 9)*, 2015.

[41] Zequn Huang, Chien-Chung Shen, Sheetal Doshi, Nimmi Thomas, and Ha Duong. Difficulty-level metric for cyber security training. In *2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2015.

[42] Nwokedi Idika and Bharat Bhargava. Extending Attack Graph-Based Security Metrics and Aggregating Their Application. *IEEE Trans. Dependable Secur. Comput.*, 9(1):75–85, January 2012.

[43] Kyle Ingols, Matthew Chu, Richard Lippmann, Seth Webster, and Stephen Boyer. Modeling Modern Network Attacks and Countermeasures Using Attack Graphs. In *Proceedings of the 2009 Annual Computer Security Applications Conference*, ACSAC '09, pages 117–126, Washington, DC, USA, 2009.

[44] Cynthia E. Irvine., Michael F. Thompson, and Ken Allen. Cyberciege: Gaming for information assurance. *IEEE Security and Privacy*, 3(3):61–64, May 2005.

[45] Isaca. *Advanced Persistent Threats: How to Manage the Risk to Your Business*. ISA, 2013.

[46] Sushil Jajodia. Topological Analysis of Network Attack Vulnerability. In *Proceedings of the 2Nd ACM Symposium on Information, Computer and Communications Security*, ASIACCS '07, pages 2–2, New York, NY, USA, 2007.

[47] Sushil Jajodia, Peng Liu, Vipin Swarup, and Cliff Wang. *Cyber Situational Awareness: Issues and Research*. Springer, 2010.

[48] Andrew Jaquith. *Security Metrics: Replacing Fear, Uncertainty, and Doubt*. Addison-Wesley Professional, 2007.

[49] Mobin Javed and Vern Paxson. Detecting stealthy, distributed ssh brute-forcing. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 85–96, New York, NY, USA, 2013. ACM.

[50] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 128–138, New York, NY, USA, 2007. ACM.

[51] Shuyuan Jin and Daniel S. Yeung. A covariance analysis model for ddos attack detection. In *in IEEE International Conference on Communications*, 2004.

[52] Klaus Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Trans. Inf. Syst. Secur.*, 6(4):443–471, November 2003.

[53] Cengiz Kahraman, Da Ruan, and Ibrahim Dogan. Fuzzy group decision-making for facility location selection. *Inf. Sci. Inf. Comput. Sci.*, 157(1-2):135–153, December 2003.

[54] Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, and XiaoFeng Wang. Effective and efficient malware detection at the end host. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pages 351–366, Berkeley, CA, USA, 2009. USENIX Association.

[55] Alexander Kott, Cliff Wang, and Robert Erbacher. *Cyber Defense and Situational Awareness*. Springer, 2014.

[56] Michael Liljenstam, Jason Liu, David M. Nicol, Yougu Yuan, Guanhua Yan, and Chris Grier. Rinse: The real-time immersive network simulation environment for network security exercises (extended version). *Simulation*, 82(1):43–59, January 2006.

[57] R. P. Lippmann and K. W. Ingols. An annotated review of past papers on attack graphs. Technical report, MIT Lincoln Laboratory, 2005.

[58] Yali Liu, Cherita Corbett, Ken Chiang, Rennie Archibald, Biswanath Mukherjee, and Dipak Ghosal. Detecting sensitive data exfiltration by an insider attack. In *Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead*, CSIIRW '08, pages 16:1–16:3, New York, NY, USA, 2008. ACM.

[59] Samuel Mahoney, Emilie Roth, Kristin Steinke, Jonathan Pfautz, Curt Wu, and Mike Farry. A Cognitive Task Analysis for Cyber Situational Awareness. In *In Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2010.

[60] Pratyusa Manadhata, Jeannette M. 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*, QoP '06, pages 3–10, New York, NY, USA, 2006.

[61] Pratyusa K. Manadhata and Jeannette M. Wing. An Attack Surface Metric. *IEEE Trans. Softw. Eng.*, 37(3):371–386, May 2011.

[62] V. Mancuso, D. Minotra, N. Giacobe, M. McNeese, and M. Tyworth. idsNETS: An Experimental Platform to Study Situation Awareness for Intrusion Detection Analysts. In *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, 2012.

[63] Microsoft. *SysInternals Utilities Suite*.

[64] MicroSoft. *MicroSoft MSDN. Windows hook functions*, 2015.

[65] Seyed Ali Mirheidari, Sajjad Arshad, and Rasool Jalili. Alert correlation algorithms: A survey and taxonomy. In *Cyberspace Safety and Security*, volume 8300 of *Lecture Notes in Computer Science*, pages 183–197. Springer International Publishing, 11 2013.

[66] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Comput. Surv.*, 46(4):55:1–55:29, March 2014.

[67] Peng Ning, Yun Cui, and Douglas S. Reeves. Analyzing intensive intrusion alerts via correlation. Technical report, Raleigh, NC, USA, 2002.

[68] Steven Noel and Sushil Jajodia. Metrics Suite for Network Attack Graph Analytics. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, CISR '14, pages 5–8, New York, NY, USA, 2014. ACM.

[69] Cyril Onwubiko and Thomas Owens. *Situational Awareness in Computer Network Defense: Principles, Methods and Applications*. IGI Global, 2012.

[70] Open Source Security. *OSSEC Manual. OSSEC V2.7.0 Documentation*, 2013.

[71] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A Scalable Approach to Attack Graph Generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 336–345, New York, NY, USA, 2006. ACM.

[72] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. MulVAL: A Logic-based Network Security Analyzer. In *Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14*, SSYM'05, pages 8–8, Berkeley, CA, USA, 2005. USENIX Association.

[73] Susmit Panjwani, Stephanie Tan, and Keith M. Jarrin. An experimental evaluation to determine if port scans are precursors to an attack. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, DSN '05, pages 602–611, Washington, DC, USA, 2005. IEEE Computer Society.

[74] Vicente Pastor, Gabriel Diaz, and Manuel Castro. State-of-the-art simulation systems for information security education, training and awareness. In *2010 IEEE Education Engineering (EDUCON)*, 2010.

[75] Cynthia Phillips and Laura Painton Swiler. A Graph-based System for Network-vulnerability Analysis. In *Proceedings of the 1998 Workshop on New Security Paradigms*, NSPW '98, pages 71–79, New York, NY, USA, 1998. ACM.

[76] Prashanth Rajivan. CyberCog:A Synthetic Task Environment for Measuring Cyber Situation. Master's thesis, Arizona State University, Tempe, AZ, USA, 2011.

[77] Prashanth Rajivan, Michael Champion, Nancy J. Cooke, Shree Jariwala, Genevieve Dube, and Verica Buchanan. Effects of Teamwork versus Group Work on Signal Detection in Cyber Defense Teams. In *In Proceedings of HCI (24)*, 2013.

[78] Mudassar Raza, Muhammad Iqbal, Muhammad Sharif, and Waqas Haider. A survey of password attacks and comparative analysis on methods for secure authentication. pages 439–444. World Applied Sciences Journal, 2012.

[79] Thomas L. Saaty and Kirti Peniwati. Group decision making: Drawing out and reconciling differences. In *RWS Publications*, 2008.

[80] Reza Sadoddin and Ali Ghorbani. Alert correlation survey: Framework and techniques. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, PST '06, pages 37:1–37:10, New York, NY, USA, 2006. ACM.

[81] SANS Technology Institute. *Assessing Outbound Traffic to Uncover Advanced Persistent Threat*.

[82] Zainab Saud and Hasan M. Islam. Towards proactive detection of advanced persistent threat (apt) attacks using honeypots. In *Proceedings of the 8th International Conference on Security of Information and Networks*, SIN '15, pages 154–157, New York, NY, USA, 2015. ACM.

[83] Karen Scarfone and Peter Mell. An Analysis of CVSS Version 2 Vulnerability Scoring. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pages 516–525, Washington, DC, USA, 2009. IEEE Computer Society.

[84] W.J. Schepens and J.R. James. Architecture of a Cyber Defense Competition. In *In Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, 2003.

[85] W.J. Schepens and J.R. James. Architecture of a cyber defense competition. In *in Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, 2003.

[86] Vyas Sekar, Nick Duffield, Oliver Spatscheck, Jacobus Merwe, and Hui Zhang. Lads: Large-scale automated ddos detection system. In *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, ATEC '06, pages 16–16, Berkeley, CA, USA, 2006. USENIX Association.

[87] 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*, SP '02, pages 273–, Washington, DC, USA, 2002. IEEE Computer Society.

[88] Oleg Mikhail Sheyner. *Scenario Graphs and Attack Graphs*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2004.

[89] Snort.org. *The Snort Project. SNORT Users Manual 2.9.3.*, 2013.

[90] John R. Surdu, John M. D. Hill, Scott Lathrop, and Curt A. Carver. Military Academy Attack/Defense Network Simulation. In *Advanced Simulation Technology Conference: Symposium on Military, Government, and Aerospace Simulation*, 2003.

[91] Colin Tankard. Advanced persistent threats and how to monitor and deter them. *Network Security*, DOI: 10.1016/S1353-4858(11)70086-1, 2011.

[92] Gianni Tedesco and Uwe Aickelin. Real-time alert correlation with type graphs. *CoRR*, abs/1004.4089, 2010.

[93] Geethapriya Thamilarasu, Sumita Mishra, and Ramalingam Sridhar. A cross-layer approach to detect jamming attacks in wireless ad hoc networks. In *Proceedings of the 2006 IEEE Conference on Military Communications*, MILCOM'06, pages 753–759, Piscataway, NJ, USA, 2006. IEEE Press.

[94] M. Tyworth, N.A. Giacobe, V. Mancuso, and C. Dancy. The distributed nature of cyber situation awareness. In *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2012.

[95] Michael Tyworth, Nicklaus A. Giacobe, and Vincent Mancuso. Cyber situation awareness as distributed socio-cognitive work. In *in Cyber Sensing*, 2012.

[96] Alfonso Valdes and Keith Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, RAID '00, pages 54–68, London, UK, UK, 2001. Springer-Verlag.

[97] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. Taxonomy and survey of collaborative intrusion detection. *ACM Comput. Surv.*, 47(4):55:1–55:33, May 2015.

[98] Vilhelm Verendel. Quantified Security is a Weak Hypothesis: A Critical Survey of Results and Assumptions. In *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*, NSPW '09, pages 37–50, New York, NY, USA, 2009.

[99] G. Vigna. Teaching network security through live exercises. In *in Cynthia Irvine and Helen Armstrong, Security Education and Critical Infrastructures*, 2003.

[100] Giovanni Vigna. Teaching network security through live exercises. In Cynthia Irvine and Helen Armstrong, editors, *Security Education and Critical Infrastructures*, pages 3–18. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[101] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An Attack Graph-Based Probabilistic Security Metric. In *Proceeedings of the 22Nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 283–296, Berlin, Heidelberg, 2008. Springer-Verlag.

[102] Weijie Wang, Baijian Yang, and Yingjie Victor Chen. Detecting subtle port scans through characteristics based on interactive visualization. In *Proceedings of the 3rd Annual Conference on Research in Information Technology*, RIIT '14, pages 33–38, New York, NY, USA, 2014. ACM.

[103] Wikipedia.org. *Advanced persistent threat*, 2015.

[104] Wikipedia.org. *Cain Abel*, 2015.

[105] Wikipedia.org. *RDP*, 2015.

[106] Wikipedia.org. *Secure Shell*, 2015.

[107] Wikipedia.org. *Stuxnet*, 2015.

[108] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '05, pages 46–57, New York, NY, USA, 2005. ACM.

[109] Zeinab Zali, Masoud Reza Hashemi, and Hossein Saidi. Real-time intrusion detection alert correlation and attack scenario extraction based on the prerequisite-consequence approach. In *In: The ISC International Journal of Information Security*, 2013.

[110] Su Zhang, Doina Caragea, and Xinming Ou. An empirical study of using the national vulnerability database to predict software vulnerabilities. In *22nd International Conference on Database and Expert Systems Applications*, Toulouse, France, 2011.

[111] Bin Zhu and Ali A. Ghorbani. Alert correlation for extracting attack strategies. In *International Journal of Network Security*, page 244258, 2006.

Appendix A

CYBER SITUATION AWARENESS SAGAT QUESTIONNAIRE

In order to evaluate the usability of the training system and the effectiveness of training, Situation Awareness Global Assessment Technique (SAGAT) is used. SAGAT covers the three levels of CSA including Level 1 (perception of data), Level 2 (comprehension of meaning), and Level 3(projection of the near future). Typically, a set of CSA queries regarding the current situation is asked and participants are required to answer each query based upon their knowledge and understanding of the situation at that point. The questions to be asked are shown in following:

1. In this scenario, the IP Address 10.7.200.2 represents
   - a Web Server
   - a Mail Server
   - a DNS Server
   - a FTP Server
2. A reasonable value for a NESSUS scan (summation of CVSS base scores for a single host) should be around
   - 0
   - between 0 and 100
   - larger than 100
   - None of the above
3. Can the outside user access the Nagios Server directly at 10.7.100.202?

- Yes
- No

4. DNC query to www.techcrunch.com is malicious or benign?
    - Malicious
    - Benign
5. If 10.7.100.102 and 10.7.100.205 were to communicate directly with each other, this behavior would be allowed or not?
    - Allowed
    - Not allowed
6. High volumes of traffic between hosts are allowed between:
    - Between 10.5.2.5 and 10.7.100.205
    - Between 10.5.1.3 and 10.5.2.4
    - Between 10.5.2.3 and 10.7.100.2
    - None of the above
7. Between 10.5.2.5 and DNS Server 10.7.100.2, there have 20 DNS query in the past 10 minutes, is it normal or not?
    - Normal
    - Not normal
8. An Snort IDS alert have been reported based on the connection from Nagios Server at 10.7.100.202 to DHCP Server at 10.7.100.3. Should this alert be reported?
    - No, it might be just regular network scanning activity
    - Yes, it might be a denial of service attack

- Yes, it might be a port scanning attack
- None of the above

Appendix B

ANALYSIS OF RECENT SOPHISTICATED ADVANCED PERSISTENT THREATS

In order to better understanding Advanced Persistent Threats (APTs), we analyzed the most recent sophisticated APTs that have huge impact. The specific details of each of these APTs attack are elaborated in the following. After analyzing all the APTs, traffic-based APT detection mechanisms are identified. In particular, the major focus of traffic-based analysis is to detect C&C traffic, as many APTs require C&C to configure and plan their sophisticated attacks

Cosmic Duke - Cosmic Duke is an information-stealing Trojan that includes code from both the Mini Duke APT Trojan, and Cosmu. Data is exfiltrated through various types of network connections, which include uploading data via FTP and three various variants of HTTP communication mechanisms. Credentials and other FTP details were found to be contained within the malware samples analyzed by F-Secure.

Mini Duke - Mini duke is a group of actors conducting a campaign of espionage using custom malware. These actors used an Adobe Reader 0-day to infiltrate target systems. Mini duke also employs a very small backdoor program written in assembly. This backdoor checks if the system is a relevant target, and then proceeds to communicate with C2 servers. Instructions are spread via twitter for further malware downloading. This malware has also been used as a base for the Cosmic Duke malware campaigns.

Dark hotel - Dark hotel is a name given to a group of threat actors that have targeted high profile targets via spear-phishing and P2P networks primarily during the target's hotel stay. They have also been known to use kernel-level key loggers, and zero-day exploits to infiltrate target systems.

Kimsuky - Kimsuky is focused solely on South Korean entities. This backdoor is believed to be spread via spear-phishing emails to vulnerable Windows systems. After initial

infection a Trojan dropper acts as a loader for other malware. As part of the infection process the software disables the firewall and the Windows Security Service Center notifications. It has a rudimentary keystroke logger but its main goal is to steal HWP documents, a similar format to MS Word but supported by a South Korean word processing application. It also downloads a modified version of a legitimate Remote Administration Tool, TeamViewer, which can grant the attacker deeper control of the infected system.

Stuxnet - Stuxnet is a nation-state backed cyber-attack campaign specifically targeting the Iranian nuclear fuel production systems, in order to slow their efforts in creating an atomic bomb. This was reported to be a joint effort by the United States and Israel.

Epic Turla - Epic Turla is a sophisticated espionage and data theft campaign characterized by a backdoor apparently controlled by Russian actors. Targeting EU governments, embassies, and the military it also has targeted research agencies, academia and pharmaceutical corporations. It makes use of several 0-day and known exploits propagated via spear phishing e-mails and watering hole attacks to gain a foothold in the target systems and then begins installing the backdoor and other programs as needed. The backdoor can monitor for network sniffing programs and will terminate itself if these are found. It has access to huge network of hacked servers for C2 and exploitation and was still active late last year.

Crouching Yeti - Crouching Yeti uses spear phishing techniques with embedded flash exploits within PDFs, as well as trojanized software installers. Methods of exploitation are considered rudimentary for this threat, as widely available exploits (such as those available in the Metasploit framework) are used in attack campaigns. Generally, the actors focus their efforts on targeting the Industrial/machinery, Manufacturing, Pharmaceutical, Construction, Education, and Information Technology industries. The most prevalent attacks are against the Industrial/machinery sector.

Duqu - Duqu is considered as a possibly either a precursor or new generation of the Stuxnet malware. The Duqu codebase is very similar to Stuxnet. The language used to program Duqu was challenging for researchers to uncover, but has since been considered an object oriented version of C. Duqu is designed to gather data to mount an attack like Stuxnet.

Shamoon - Shamoon was a virus designed to destroy data on target hosts. Most

security experts believe that the malware was designed by politically motivated threat actors. The malware infected 30,000 Aramco computer systems, and was capable of wiping the data contained on the hard drive, as well as partition information in the MBR. The malware can then report infection information back to the attackers.

Flame - Flame is believed to be a state-run cyber espionage operation due to the sophistication and complexity of the attack, Flame may have also been contracted out by the same nation-state behind Stuxnet and Duqu. It is designed to steal documents, recorded conversations, and keystrokes, and also opens a backdoor to allow attackers to tweak and add new functionality. The tools leverage Bluetooth to collect data from nearby Bluetooth-enabled devices. It also has the ability to sniff traffic visible from the target.

NetTraveler - NetTraveler is a cyber-espionage toolkit that targets Tibetan/Uyghur activists, and more recently the space exploration, nanotechnology, energy production, nuclear power, lasers, medicine and communications industries. Thought to be of Chinese origin, the attackers gain a foothold via spear phishing campaigns with malicious MS Office documents, using two well-known (and patched) vulnerabilities. Nevertheless, these attacks were very successful, even against the high-profile targets. After a successful exploitation NetTraveler is designed to extract large amounts of private information from the victims system over long periods of time. The malware uses compression techniques and a fail-safe protocol to ensure that target data is transferred to the attackers C2 servers.

Regin - Regin is a cyber-attack platform, thought to be created and controlled by a nation-state actor, which the attackers deploy in the victim networks for ultimate remote control at all possible levels. It targeted telecoms, government entities, multi-national political bodies, financial institutions, academia/research, specific individuals involved in cryptographic research in Russia, Central Asia and other parts of the world. Its main purpose was intelligence gathering and facilitating other forms of attack across networks. It was also the first cyber-attack platform known to penetrate and monitor GSM networks in addition to its other abilities. It is a modular, sophisticated platform with very advanced capabilities.

TeamSpy - From 2004 to 2013 the TeamSpy group of actors used a malicious version

of TeamViewer, a legitimate Remote Administration Tool to monitor activists and steal sensitive data, crypto keys and passwords and other data from intelligence agencies and heavy industry manufacturers in the Commonwealth of Independent States (CIS) and other Eastern European nations. This covert cross-nation, cyber surveillance data theft and monitoring operation made use of legitimate, signed software packages in addition to custom made software, including DLL path hijack tricks and allowed the threat actor to conduct effective operations targeting hundreds of victims, including high level/high value individuals. Although it was not a particularly advanced campaign, using simple encryption techniques and lazy software choices, it was very effective for many years.

Wiper - In 2012 businesses in Iran specializing on energy, oil and gas production as well as government entities were reportedly victims of a data wiping program known as Wiper. This program was so well written that no known copies were ever recovered; once activated no data that could be used to recreate it survived. Its purpose was solely to destroy data on hard disks using an extremely efficient wiping algorithm. There is some thought that it was an extension of the Stuxnet or Duqu code but this cannot be ascertained.

Winnti - Winnti is a cybercriminal entity waging an IP theft campaign against online gaming companies and game developers in Southeast Asia and other areas. Using a malicious DLL to infect a companys servers gives the actors access to potentially millions of computers but the main aim was to steal intellectual property, digital certificates, source code and system design. During the course of the campaign other Remote Administration Tools are downloaded as needed.

SabPub - SabPub is a backdoor targeting the Dalai-Lama and activists in the Tibetan community on the OS X platform. It is almost certainly of Chinese origin. It uses social engineering to get users to open exploited MS Word documents and install the software. A second version was created that uses Java exploits to infect target machines.

FinSpy - FinSpy, created by Gamma International (UK), is another example of Business-to-government malware marketed towards government and law enforcement agencies for the purpose of monitoring criminals and activists. It has characteristics of several different types

of malware including backdoors, Trojans, rootkits and bootkits. It has several versions targeting nearly every platform an available; Windows, OS X, Linux, Android, iOS, Windows Mobile, Symbian and BlackBerry. It was developed to focus mainly on smartphones and other mobile devices and gives its controllers the ability to log incoming and outgoing calls, makes concealed calls to eavesdrop on the target's surroundings, steal information (call logs, text and media messages, and contacts) and track coordinates.

BlackEnergy - First observed back in 2007, BlackEnergy was originally a simple DDoS Trojan that has become a much more extensive tool capable of sending spam and aiding in the committal of banking fraud. It also increases the feasibility of more advanced targeted attack activities. In 2010, the second version of BlackEnergy was observed to feature rootkit techniques. Recent targeted attacks using this Trojan have created awareness in the fact that this is an active force requiring defensive action.

Hacking Team RCS - Remote Control System (RCS) is a spyware program developed by the Italian company HackingTeam specifically for sale to law enforcement and government agencies. RCS not only has backdoor, rootkit and Trojan features but can compromise a variety of systems and operating systems including Windows, OS X, BlackBerry, Windows Mobile, Android and iOS. It is a self-replicating program designed to steal and exfiltrate personal data to a remote server. It intercepts data from web browsers, email clients and instant messaging services as well as video and audio streams and geolocation data. It utilizes known exploits as well as 0-day code for Microsoft Office documents, Adobe Flash and others.

Madi - Madi is a Trojan campaign directed at critical infrastructure engineering firms, government agencies, financial organizations, and academia in the Middle East and Israel. Cyber-espionage oriented, it used relatively simple social engineering approaches to get victims to install the malicious code. Religious and political distraction documents and images were used to trick users. Programming was apparently written by inexperienced programmers using Delphi and applying rudimentary approaches; no 0-day or elegant exploits were used. After clicking on what appeared to be a PPT or jpg file the Trojan automatically installed and provided several functions to controllers including; key logging, file retrieval,

retrieve disk structures, record audio and screenshot capture for specific events.

Machete - Machete was a cyber-espionage/data theft campaign targeted at South American embassies, government organizations and intelligence agencies from 2013 to 2014, that may still be active. Using known exploits for Windows Office documents via a dedicated spear phishing campaign and a fake blog threat actors are able to install a multi-function program that can log keystrokes, capture screenshots, web cam photos and microphone audio, geolocation data and copying files to USB drives and remote servers. It appears the attackers used Python embedded into Windows executables for ease of coding.

Cloud Atlas - Cloud Atlas is a rebirth of the Red October series of attacks. After discovery by Kaspersky in 2013 the Red October campaign was promptly shuttered and the group vanished for several months. Cloud Atlas shares several similarities with the Red October campaign, including the same types of targets (in some instances the exact same individuals were targeted), shellcode markers in malicious documents, compression algorithm and C&C servers. In both cases the top 5 countries targeted matched exactly and some of the spear phishing documents had the same name. The Cloud Atlas campaign is currently still active.

Red October - Red October was a series of attacks against diplomatic, governmental and scientific research targets in Eastern Europe and Central Asia. Actors focused on gathering intelligence from target organizations it began as early as 2007 and lasted until 2013. Initial infection was enabled via known exploits for MS Office documents and Adobe PDF files. Attackers crafted a multi-function kit that enabled them to steal data from not only workstations but smartphones, Cisco network devices and removable disk drives. This particular kit had more than 30 modules the attackers could leverage.

IceFog - When discovered in 2013 Icefog represented a relatively new trend; small groups of highly skilled attackers targeting supply chain targets. Icefog targeted government institutions, military contractors, maritime and ship-building groups, telecom operators, satellite operators, as well as industrial and high technology companies in South Korea and Japan for cyber-espionage. Using custom-made tools for Mac OS X and MS Windows, they relied on spear-phishing and exploits for known vulnerabilities to initially infect their

victims. After gaining a foothold they moved laterally throughout the victims network and focused on very specific data; company plans, sensitive documents and the like. After exfiltrating the data they abandoned the infection and moved on.

Appendix C

APT DETECTION SYSTEM RUN-TIME DATA COLLECTION

C.1 NetFlow Data Format for APT Detection

In APT reconnaissance stage, most of the abnormal behaviors can be feasibly detected by NetFlow-based traffic analysis. Originally developed by Cisco Systems, NetFlow is an open source traffic profile monitoring technology and a de facto industry standard to describe the method for a router to export statistics about the routed socket pairs. When NetFlow is enabled on a router interface, traffic statistics of packets received on that interface will be counted as flow and stored into a dynamic flow cache. There is no packet payload information in the flow field, and it is one of the major differences between NetFlow and the traditional IDS. Paying no attention to packet payloads greatly reduces the processing overhead and makes NetFlow an extraordinarily good fit for real-time analysis of high-volume network traffic in large-scale operational environments. Be default, NetFlow records can be exported to a user-specified monitoring station using UDP packets, if one of the following conditions occurs:

- The transport protocol indicates that the connection is completed (TCP FIN), and there is a small delay to allow for the completion of the FIN acknowledgment handshaking;
- Traffic inactivity exceeds 15 seconds;
- For flows that remain continuously active, flow cache entries expire every 30 minutes to ensure periodic reporting of active flows.

Note that the NetFlow record represents the packets in an aggregated format. Each line does not represent the information of a packet. Instead, each line represents the information of all packets in a flow (which is defined as a series of packets sent from a source IP

address to a destination IP address over a period of time). The NetFlow data format is show in the following:

Table C.1: NetFlow data format

Column number	Contents	Description
1	start	Flow start time
2	start_msec	Residual milliseconds of flow start time
3	end	Flow end time
4	end_msec	Residual milliseconds of flow end time
5	srcaddr	Source IP address
6	srcport	Source port number
7	src_as	Source AS
8	dstaddr	Destination IP address
9	dstport	Destination port number
10	dst_as	Destination AS
11	prot	Protocol
12	tcp_flags	TCP flags
13	tos	Type of service
14	dPkts	Number of packets
15	dOctets	Number of bytes
16	nexthop	Next hop IP address (0 if null)
17	input	Input Interface
18	output	Output interface
19	engine_id	Switching Engine ID

C.2 Linux System Resource Exposure Sensor Record Normalization for APT Detection

As part of the APT detection system, we developed the auditing sensor in the Linux platform. The auditing sensor resides at each host to monitor the resource access activities and interactions of all processes on the host. The sensor monitors how critical resources such as files, registries, and network sockets are exposed to the target process. Furthermore, the sensor collects and combines various heuristics such as processes accessing files in other users folders and system directories, processes modifying critical registries, creating a large number of sockets during a short period of time.

To comprehensively log and track access to files, directories, and resources of Linux system, as well as trace system calls, we leverage the Linux Auditing System called auditd. By creating a sophisticated set of rules including file watches and system call auditing, the sensor will be able to report suspicious activities that violating the security policies, for example, tracking any unauthorized change in critical system configuration files such as /etc/passwd. Then, the audit records are analyzed periodically, normalized, and streamed through Kafaka messaging system in real-time.

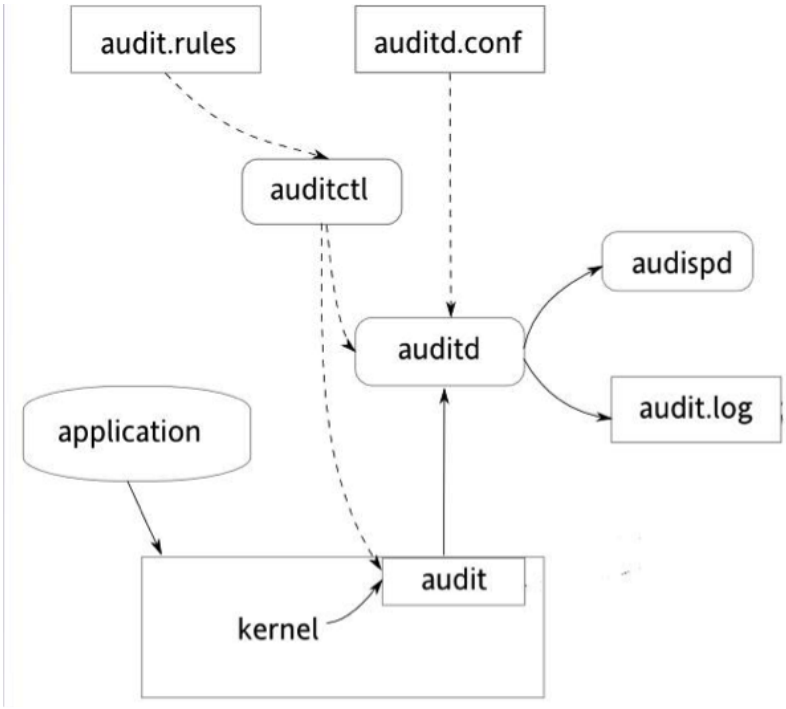


Figure C.1: Linux auditing system architecture

The auditing sensor system architecture is illustrated in Figure C.1. Auditd is auditing daemon located in the kernel that keeping monitoring the applications behavior. The behavior of the auditd server can be controlled through auditctl tool dynamically. With auditctl, we can turn auditing on or off, check the status and add audit rules for specific events. Audisp is the Audit dispatcher daemon interacts with the Audit daemon and sends events in real-time through Kafka messaging system for further processing.

To use auditing sensor, use the following steps:

1. Configure the audit daemon.

2. Add audit rules and watches to collect desired data such as to monitor certain system calls as well as monitor the read, write, and execute change on some system critical files and folders.

3. Start the daemon, which enables the Linux Auditing System in the kernel and starts the logging.

4. Periodically analyze data by normalizing audit record into object-oriented format.

5. Stream the normalized records through Kafaka messaging system in real-time for future processing.

```
1 type=SYSCALL msg=audit(1439833529.362:1438): arch=c000003e syscall=82 success=no
  exit=-2 a0=1177b60 a1=1177b90 a2=0 a3=6f6c2e726567616e items=2 ppid=3088
  pid=3093 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=104 sgid=0
  fsgid=104 tty=(none) ses=4294967295 comm="logrotate" exe="/usr/sbin/logrotate"
  key=(null)
2 type=CWD msg=audit(1439833529.362:1438):  cwd="/var/lib/logrotate"
3 type=PATH msg=audit(1439833529.362:1438): item=0 name="/var/log/upstart/"
  inode=24911 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00 nametype=PARENT
4 type=PATH msg=audit(1439833529.362:1438): item=1 name="/var/log/upstart/"
  inode=24911 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00 nametype=PARENT
```

Figure C.2: Linux auditing log

The above auditing log event shown in Figure C.2 consists of four records (each starting with the type= keyword), which share the same time stamp and serial number. Each record consists of several name=value pairs separated by a white space or a comma. A detailed description of log file attributes are elaborated in Table C.2.

Since the APT detection system collecting alerts from different sensors and these alerts are encoded in different formats, we have to translate all attributes of each sensor alert into a common format. This translation requires that the syntax and semantics of a sensor alert are recognized. In order to express the causal dependency relationship between two collected system objects, such as processes, connections, files, etc, the alerts are normalized into the Source Object, the Sink Object and the Action format. This kind of object-oriented

Table C.2: Linux auditing log attribute

Name	Description
type	The type field contains the type of the record, which may be SYSCAL, CWD, PATH.
msg	The msg field records include a time stamp and a unique ID of the record.
arch	The arch field contains information about the CPU architecture of the system.
syscall	The syscall field records the type of the system call that was sent to the kernel.
success	The success field records whether the system call recorded succeeded or failed.
exit	The exit field contains a value that specifies the exit code returned by the system call.
a0a3	The a0 to a3 fields record the first four arguments of the system call in this event.
items	The items field contains the number of path records in the event.
ppid	The ppid field records the Parent Process ID (PPID).
pid	The pid field records the Process ID (PID).
auid	The auid field records the Audit user ID, that is the loginuid.
uid	The uid field records the user ID of the user who started the analyzed process.
gid	The gid field records the group ID of the user who started the analyzed process.
euid	The euid field records the effective user ID who started the analyzed process.
suid	The suid field records the set user ID of the user who started the analyzed process.
fsuid	The fsuid field records the file system user ID of the user who started the process.
egid	The egid field records the effective group ID of the user who started the analyzed process.
sgid	The sgid field records the set group ID of the user who started the analyzed process.
fsgid	The fsgid field records the file system group ID of the user who started the process.
tty	The tty field records the terminal from which the analyzed process was invoked.
ses	The ses field records the session ID of the session.
comm	The comm field records the command-line name of the command.
exe	The exe field records the path to the executable used to invoke the analyzed process.
key	The key field records the administrator-defined string associated with the rule.
cwd	The cwd field records the current working directory

design is able to express relationships between alerts which is an essential requirement of alert correlation.

After aggregating the auditing log records with same serial number, the records are normalized into Source Object, Action, and Sink Object with json format as shown in Figure C.3 and Figure C.4. Notice that Figure C.3 illustrates the audit record when unauthorized rm action was performed on the monitored directory /root/test. Figure C.4 shows the audit record when an outside connection to IP address 116.109.112.47 and port 47 is captured by the audit daemon.

Figure C.3: Normalized Linux audit record of removing file

Figure C.4: Normalized Linux audit record of Internet connection

C.3 Windows System Resource Exposure Sensor Record Normalization for APT Detection

Security auditing is a powerful tool to help maintain the security of a system. Windows security and system logs record the potentially harmful behaviors, such as changes to user account and resource permissions, failed attempts by users to log on, failed attempts to access resources, and changes to system files. Windows auditing system also provide options to monitor the read, write, and execute change on some system critical files, folders,

and registry.

Windows auditing system also provide options to monitor the read, write, and execute change on some system critical files and folders. The auditable actions for files and folders are described in Table 4. In order to turn on these audit actions, you have to right-click the document or file that need to be kept track of, and then click Properties, click the Security tab, click Advanced, and then click the Auditing tab. Then choose the corresponding audit actions. The auditable actions for files and folders are described in Table C.3

Table C.3: Windows auditing log attribute

Name	**Description**
Traverse folder/execute file	Keeps track of when someone runs a program file.
List folder or read data	Keeps track of when someone views the data in a file.
Read attributes	Keeps track of when someone views the attributes.
Read extended attributes	Keeps track of when someone views the extended attributes.
Create files or write data	Keeps track of when someone changes the contents of a file.
Create folders or append data	Keeps track of when someone adds data to the end of a file.
Write attributes	Keeps track of when someone changes the attributes of a file.
Delete subfolders and files	Keeps track of when someone deletes a folder.
Delete	Keeps track of when someone deletes a file.
Read permissions	Keeps track of when someone reads the permissions on a file.
Change permissions	Keeps track of when someone changes the permissions on a file.
Take ownership	Keeps track of when someone takes ownership of a file.

For a single operation such as delete an audited file, it will generate three events: Windows 4656, Windows 4663, and Windows 4658. These three events can be associated through the same HandleId.

1. Windows 4656: A handle to an object was requested.

   When enable auditing on an object (e.g. file or folder), this is the first event recorded when an application attempts to access the object in such a way that matches the audit policy defined for that object in terms of who is requesting the access and what type of access is being requested.

2. Windows 4663: An attempt was made to access an object

This event documents actual operations performed against files and other objects. It is logged between the open (4656) and close (4658) events for the object being opened and can be correlated to those events via Handle ID. While event 4656 tells you when the object is initially opened and what type of access was requested at that time; 4656 doesn't give you positive confirmation any of the access permissions were actually exercised.

3. Windows 4658: The handle to an object was closed

   After successfully opening an object, a program eventually closes it.

Name	**Description**
Mission	A combination of tasks to achieve a common goal.
Task	A usually assigned piece of work often to be finished within a certain time.
Activity	A unit, organization, or installation performing a function or mission, e.g., reception center, redistribution center, naval station, naval shipyard.
Essential Task	A specified or implied task that an organization must perform to accomplish the mission. An essential task is typically included in the mission statement.
Specified Task	In the context of joint operation planning, a task that is specifically assigned to an organization by its higher headquarters.
Implied Task	In the context of joint operation planning, a task derived during mission analysis that an organization must perform or prepare to perform to accomplish a specified task or the mission, but which is not stated in the higher headquarters order.

Cyber Capability	An ability to execute a specified course of action. A combination of network services and assets that provide users with the ability to perform an action. The term “cyber capability” is at a higher level of abstraction than device, asset, service, or resource. It expresses an operational need in cyber terms. Examples: Near real-time communication is a cyber capability enabled by VoIP, instant messaging, email, and texting. File transfer is a cyber capability enabled by FTP, email attachment, or instant message attachment.
Site	An installation, together with its personnel and equipment, organized as an operating entity.
User	Individual or (system) process authorized to access an information system.
Cyber Asset	Programmable electronic devices and communication networks. It includes hardware (e.g. workstations, servers, switches), software (e.g. operating systems, applications), and data (e.g. document files, images). A broad term that denotes something of value on a computer network.
Cyber Resource	A general term that incorporates a wide variety of components of a cyber infrastructure. It includes cyber assets, network services, storage media, and,physical connections.
Network Service	Combination of one or more ports open to incoming connections to provide a service to clients. Examples: email, printing, Domain Name Service (DNS), File Transfer Protocol, etc.
Node	In network topology, a terminal of any branch of a network or an interconnection common to two or more branches of a network.
Host	Almost any kind of computer, including a centralized mainframe that is a host to its terminals, a server that is host to its clients, or a desktop personal computer (PC) that is host to its peripherals. In network architectures, a client station (user’s machine) is also considered a host because it is a source of information to the network.

Network Device	Network device refers to an active device on the network that connects or manages network traffic, such as repeaters, hubs, switches, bridges, routers, and gateways. Hardware or software (virtual machine) with an IP address. It is a type of cyber asset.
Application	Software program that performs a specific function directly for a user and can be executed without access to system control, monitoring, or administrative privileges.
Data	Distinct pieces of digital information that have been formatted in a specific way.
File	A collection of information logically grouped into a single entity and referenced by a unique name, such as a filename.
Process	A process is an instance of a computer program that is being executed. It contains the program code and its current activity

Table C.5: Host information ontology

The default audit event is shown in Figure C.5

In order to express the causal dependency relationship between two collected system objects, such as processes, connections, files, etc., the alerts are normalized into the Source Object, the Sink Object and the Action as we did in Linux auditing sensor. Figure C.6 shows the normalized windows audit record when an unauthorized DELETE action was performed on the monitored directory.

```
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event"><System><Provider
Name="Microsoft-Windows-Security-Auditing" Guid="54849625-5478-4994-a5ba-
3e3b0328c30d"></Provider>
<EventID Qualifiers="">4663</EventID>
<Version>0</Version>
<Level>0</Level>
<Task>12800</Task>
<Opcode>0</Opcode>
<Keywords>0x8020000000000000</Keywords>
<TimeCreated SystemTime="2015-08-10 20:30:59.139915"></TimeCreated>
<EventRecordID>1233631</EventRecordID>
<Correlation ActivityID="" RelatedActivityID=""></Correlation>
<Execution ProcessID="4" ThreadID="48"></Execution>
<Channel>Security</Channel>
<Computer>IAI162.iai.pri</Computer>
<Security UserID=""></Security>
</System>
<EventData><Data Name="SubjectUserSid">S-1-5-21-880973042-2781994814-3549891983-
4669</Data>
<Data Name="SubjectUserName">zhuang</Data>
<Data Name="SubjectDomainName">IAI</Data>
<Data Name="SubjectLogonId">0x000000000003f303</Data>
<Data Name="ObjectServer">Security</Data>
<Data Name="ObjectType">File</Data>
<Data Name="ObjectName">C:\Test</Data>
<Data Name="HandleId">0x0000000000001454</Data>
<Data Name="AccessList">%%4416
        </Data>
<Data Name="AccessMask">0x00000001</Data>
<Data Name="ProcessId">0x0000000000000884</Data>
<Data Name="ProcessName">C:\Windows\explorer.exe</Data>
</EventData>
```

Figure C.5: Windows audit record of deleting file

Table C.4: APT detection system event object ontology

File	SequenceNumber UtcTime ProcessGuid ProcessId Image TargetFilename CreationUtcTime PreviousCreationUtcTime
Process	SequenceNumber UtcTime ProcessGuid ProcessId Image CommandLine CurrentDirectory User LogonGuid LogonId TerminalSessionId IntegrityLevel Hashes ParentProcessGuid ParentProcessId ParentImage ParentCommandLine
Network Connection	SequenceNumber UtcTime ProcessGuid ProcessId Image User Protocol Initiated SourceIsIpv6 SourceIp SourceHostname SourcePort SourcePortName DestinationIsIpv6 DestinationIp DestinationHostname DestinationPort DestinationPortName

Figure C.6: Normalized Windows audit record of deleting file