A Methodology to Identify Complex Network Attacks

Lisa Frye[1,2], Liang Cheng[2], and Randy Kaplan[1]
[1]Computer Science Department, Kutztown University, Kutztown, PA, USA
[2] Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, USA

Abstract—*Networks are attacked on a daily basis. Identifying these attacks is a crucial task for network managers. Tools exist to assist in this task. One of the more common tools used is an Intrusion Detection System. These systems may fall short in identifying all attacks that have occurred on a network. Often an attack consists of a sequence of simple attacks. These complex attacks are more difficult to identify. The proposed methodology identified a family of complex attacks to aid in the understanding of these attacks to improve their detection. A reusable representation of these attacks was developed using ontology. Lastly, an algorithm was developed to utilize the ontological representations to extend the knowledge of complex attacks and allow for better detection of such attacks by taking advantage of the advanced reasoning inherit in ontology.*

Keywords: Computer network security, Nonmonotonic reasoning, Security, Site security monitoring

1 Introduction

Networks are common in everyday life. Networks consist of many nodes with a plethora of data traversing the network daily. This data consists of legitimate data for professional and personal purposes, as well as undesirable data, often injected willingly by users intending to attack the network or a node on the network. The intention of the attack may vary from the user that just wants to prove the attack can occur, to more malicious attacks intending to disrupt services or destroy data. It is important to be able to identify when an attack occurs, stop any damage being caused by the attack and implement additional security measures to prevent such attacks in the future.

One tool often used to identify network attacks is an Intrusion Detection System (IDS) [1]. There are many different types of IDSs, ranging from host-based (HIDS) [1], residing on a specific host watching for attacks against that host, to network-based (NIDS) [1], deployed on the network to monitor network resources and identify network attacks. Organizations typically use a combination of network-based and host-based IDSs.

IDSs can also be classified by how they detect attack attempts. A signature detection IDS [2] uses a set of rules to look for attacks. The rules contain patterns that correspond to the data that represents a possible attack. If the pattern matches data on the network or in a host there may be a possible attack in progress.

Another type of IDS, called an Anomaly Detection IDS [2], looks for abnormal network behavior to identify a possible attack. Obviously a single type of IDS will not be sufficient to detect all types of attacks. Therefore, by combining the capabilities of various IDSs a more comprehensive approach to attack detection can be deployed.

The IDSs described thus far have one characteristic in common. That characteristic is that they examine snapshots of data in order to detect an attack. A snapshot of data that is examined from a network represents a single state of an aspect of the network. If an attack is such that it can be recognized by examining a single piece of data (usually a packet) then these methods will be successful in detecting a potential network attack.

A complex network attack is one in which the temporal domain and the spatial domain of the data must be examined in concert in order to determine if an attack is underway. The temporal domain of an attack consists of examining a period of time in which network events occur. During this period of time there may be multiple events that represent an attack is in progress.

The spatial domain refers to position or location in the network. For an IDS this might mean physical location, logical location in the network, such as the subnet, or the node's neighbor nodes. The spatial domain consists of two aspects. One of these is strictly spatial and the other is a temporal-spatial aspect. The first aspect is where the events occur in the topography of the network. Again, certain events occurring at certain locations in the network may indicate an attack while those at other locations may not. The second aspect is represented by the sequence of events. In other words, a specific sequence of events during a specific time period may represent a possible attack whereas a different sequence of the same events may not.

As attacks become more complex, i.e., having several attack vectors across multiple temporal and spatial domains, more sophisticated IDSs will be necessary to detect these more sophisticated attacks. Our research involves designing a process for the detection of complex network and host attacks.

1.1 Detecting Sophisticated Network Attacks

In order to create an IDS that can detect complex network and host attacks it is necessary to employ some mechanism that can recognize an attack based on multiple events occurring in multiple locations at different times. In order to accomplish this it is necessary to be able to recognize

combinations of events that represent possible attacks.

Our research involved developing a reasoning system to monitor network traffic and identify the occurrence of a complex attack. Packets and sequences of packets were examined so a more robust analysis could be completed.

Consider a simple example of the kind of network event to observe in order to detect a complex attack attempt. One of the ways that attackers breach systems is by searching for open ports. Observing an attempt being made to determine if a specific port is open indicates very little about whether an attack is underway. Alternatively, if network traffic is observed indicating that a sequence of open port checks is being made over a period of time consistent with a port scanner, then, in fact, an attack may be in progress.

Our reasoning system consisted of a description of specific attack elements that could take place over a network. An attack element is a description of a complex attack component. This component combined with other components comprises the attack.

Decomposing attacks into attack elements enables the ability to better understand how attacks are constructed by attackers. For example, it may be the case that a certain combination of exploits is used to achieve a certain type of breach. Knowing the result of this combination provides the capability to consider other related combinations of attacks that yield the same results. Our research entailed exploring to what extent the assemblage of attack elements into potential attacks could be performed automatically. By doing so, the initial knowledge of specific attacks could be leveraged into a wider, more comprehensive set of attack descriptions.

This automation process allows for a more extensive range of attacks to be identified, including potential zero-day attacks (a new attack that exploits a vulnerability unknown to the general public or software developers). The reasoning system employs an ontological representation of attack elements and attacks. Such a representation describes entities that will be reasoned about and also how the reasoning will take place.

Our contribution to the field of intrusion detection is threefold. First, by identifying a family of complex attacks we enable better detection of these types of attacks. Second, by representing these complex attacks ontologically we create an advanced and reusable representation of network attacks. Third, by developing an algorithm that reasons about attacks using ontology [3], which we will develop, we provide a means to extend our knowledge of complex attacks allowing for better detection of such attacks.

The remainder of this paper is organized as follows. Section II discusses related work. Section III discusses the methodology for the approach developed in this research. A simulation environment is described in Section IV. Section V provides conclusions and future work.

2 Related Work

Over the years there has been a significant amount of IDS research. Many surveys of the current state of IDSs and IDS research exist, such as in [2]. The basic types of IDSs are either signature- or anomaly-based. In a signature-based IDS packet data is examined to determine if it represents a known attack pattern. In an anomaly-based IDS the behavior of the network is considered and an attempt is made to recognize aberrant network behavior(s).

The evolution of IDSs consists of more complex approaches to analyzing network data to make assessments of the state of the network to determine if there is a potential breach. More advanced techniques incorporate data mining [4], fuzzy logic [5], learning [6], and logic [7].

Huang and Wicks [8] use the analogy of an intrusion to that of a battlefield. In intrusion detection as well as in the battlefield they cite a number of shared characteristics including an environment that is heterogeneous and widely distributed, a significant amount of data that is constantly changing and which can be extremely noisy, incomplete and inconclusive information that makes decision making difficult, and attack patterns which are constantly changing. When thinking about intrusion detection, one must take these characteristics into account when devising mechanisms for detection.

Huang and Wicks [8] point out that if a file-access-violation is detected, the true purpose of this event cannot be determined without additional information referred to as context. Such contextual information would include such information as the present machine configuration, the location of the files, permissions, and account configuration. The important point that Huang and Wick make is that by the time sufficient information arrives at a central analysis point, the situation (context) may have changed drastically. Huang and Wicks approach to analyzing what may be happening it is appropriate to consider the strategy the attacker may be using. This in turn calls for a description of the attacks that more abstract in nature. This is consistent with the approach described in this paper, namely to represent descriptions of attacks in the form of a conceptual ontology.

In Camtepe and Yener [9] an approach to detection complex attacks is presented that is based on the construction of finite automatons that represent the “patterns” of complex attacks. The define a non-deterministic enhanced finite automata to be a tuple consisting of Q, a set of states, Q_{PA}, a set of partial attack states, Q_A, a set of attack states, F, the input alphabet, D, a set of derivation rules for goals and subgoals, $DELTA_F$ and $DELTA_B$, sets of forward and backward transition rules. The finite automata can recognize complex attack patterns. The automata implicitly specify the relationships between the attack elements and therefore, unlike a conceptual representation, no ability to generalize or specialize exists without the specification of another automata.

A Process Queuing System (PQS) was the method used in [10] to detect complex attacks. The complex attacks were represented as finite state machines (FSM) with the attack elements represented as states and the transitions were triggered by observations about the occurrence of an attack element or a response to an attack element. The FSM were represented as models, which could be incorporated into a hierarchy of models, allowing for high-level modesl to be

developed to detect complex attacks based on results of lower-level models.

The MulVAL [7] system uses a logical deduction process to determine the existence of an attack on the network. It consists of generic rules, including rules to determine if a vulnerability exists and the consequence of an exploit against the vulnerability. The network properties are obtained by the scanning process, consisting of scanners that run on each host and report their findings to the host running MulVAL. MulVAL then runs an analyzer on the properties received from all the scanners. This analysis is done in two phases, an attack simulation phase and a policy checking phase. The attack simulation phase identifies all possible data accesses of an attacker, which are then sent to the policy checking phase. This phase compares the output of the attack simulation phase with the specified security policy and identifies violations. The scanner must be run on each host and identifies vulnerabilities specific to each host.

An ontology-based intrusion detection system was described by Mandujano [11] and Mandujano, Galvin, and Nolazco [12]. In this approach, the authors are looking to detect outgoing intrusions. The ontology they propose enables the detection of code and network activity that identifies a possible intruder. The ontology specifies concepts like hostile and safe processes as subclasses of process for example. There ontology, unlike the ontology proposed in this paper does not distinguish between traffic and attack. It is our contention that such a distinction is necessary to successfully identify sequences of incoming attacks and also to be able to recognize the type and kind of attack that is transpiring.

Another system that makes use of ontology as its basis for operation is the Reaction after Detection (ReD) Project [13]. ReD was developed to determine the best reaction to an attack. The system architecture consists of several components that assist in deciding short and long term reactions. The long term reaction consists of the deployment of new security policies to the network. An ontology-based approach is used to instantiate these new security policies. ReD makes decisions based on existing security policies and reacts by instantiating new policies. The ontologies employed in ReD provide the basis for recognizing policy violations. Rules are employed to analyze a violation and also to determine an appropriate reaction. An attack recognition and remedy may involve additional aspects beyond security policies, which were considered in the model proposed in our research.

Undercoffer, Joshi, and Pinkston [14] described a model for a host-based IDS system that uses ontology. The anomaly detector detects abnormal behavior at the system level and determines if data samples fall within a normal state compared to a baseline. Subsequent samples that fall outside the bounds of the normal state are identified as possible abnormal behavior. The ontologies define the properties of the target of the attack and the attack itself, including the consequences and means of the attack. Once again, this system is host-based and does not consider all the network components, such as the routers and switches in the network.

Similar to many of these examples, the IDS we are proposing uses ontology to represent complex types of attacks that can take place. Unlike these examples, our system utilizes ontology to identity complex attacks against any node in the network, including network devices. Our literature survey indicates that the detection of complex attacks remains an area of research that has not reached a viable solution. Our proposed method for improving performance of IDS for detecting complex attacks represents a significant contribution to this research.

3 Methodology

The occurrence of simple attacks may indicate that an attacker is just trying an easy attack. The assemblage of several simple attacks may indicate the occurrence of a more complex attack. In order to understand the way simple attacks may fit together to form a complex attack, it is necessary to consider their spatial and temporal properties. For instance, pings to hosts on the same network, with incrementing IP addresses, over a span of several days, may indicate a network manager doing simple management or troubleshooting tasks. Given the same set of pings, over a span of several minutes, typically indicates an attacker looking for available hosts to attack. It is necessary to see that these ping packets are generated from the same source host and also within the same time period.

This research analyzed network packets captured using the Wireshark [15] packet capture software from both a spatial and temporal view. The analysis made use of some existing tools. Snort [16, 17], a combined NIDS and Intrusion Prevention System (IPS), was used to check the captured packets for possible attacks identified by the snort rules. The tcpdump [18] utility was used to convert the pcap file from Wireshark to an ASCII format for processing.

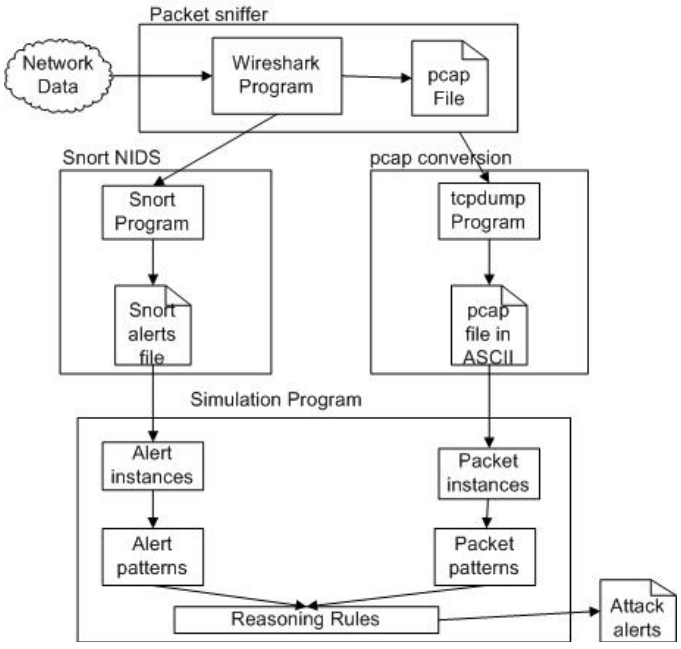

Figure 1: Reasoning system design.

The overall system design is illustrated in Fig. 1. The network data was captured using Wireshark, which created a pcap file of all the data captured. This pcap file was processed by snort and tcpdump, which generated output for use by the simulation program. The simulation program used the snort alert file, output from the snort processing, and the ASCII form of the pcap file, output from the tcpdump processing. Both of these outputs were used to create instances of the alerts and packets. These instances were used as inputs to the reasoning rules using pattern matching, which produced attack alerts.

TABLE I
PROPERTIES FOR ALERTS AND PACKETS

Property Description	Alert Field	Packet Field
Date and time of item	X	X
Item description	X	
Classification of item	X	
Source IP address	X	X
Destination IP address	X	X
Source port number	X	X
Destination port number	X	X
Source MAC address		X
Destination MAC address		X
IP version number		X
Length of IP header	X	X
Length of IP datagram	X	X
IP upper-layer protocol		X
IP flags		X
IP fragment offset		X
IP time to live		X
IP checksum		X
TCP sequence number	X	X
TCP acknowledgement number	X	X
TCP flags	X	X
TCP window size		X
Upper-layer protocol checksum		X
ICMP type	X	X
ICMP code	X	X
Application-layer protocol		X
Application-layer data (first 80 bytes)		X

The alert field and packet field columns indicate if the property is maintained for an alert or packet, respectively.

Instances were created for the alerts and packets. These represented all the alerts found by snort and all the packets captured with Wireshark. Table 1 shows the properties that were maintained for the alerts and packets. Ontologies are used to map data instances into attacks. The result of this mapping is an instance of a colored attack tree.

The instances were searched for the occurrences of patterns representing simple attacks. If a pattern was found, data corresponding to the attack was written to an output file.

A complex attack is the combination of several simple attacks. Many times it is necessary for these simple attacks to occur within a specified timeframe. To determine if the simple attacks occurred within the specified timeframe, the time from the first node being colored in an attack tree to the time the root node is colored is measured. Finding the most effective timeframe is a critical step in complex attack identification and will be determined in future work.

3.1 Attack Trees

Attack trees were created for the complex attacks used as examples (see Fig. 2). Each step was given a unique node id. For each complex attack example, data was generated as the attack was conducted. The data was manually analyzed to identify patterns so the attack could be identified from either the snort alerts or the captured data packets. A coloring scheme was used in the attack trees to identify the nodes that correspond to identified simple attacks.

Steps in a complex attack are often similar across several complex attacks. For instance, many attacks begin by the attacker identifying available hosts on a network and then proceeding by identifying open ports on each available host found. These generic simple attacks were identified with a unique attack id number. A mapping was created to map the

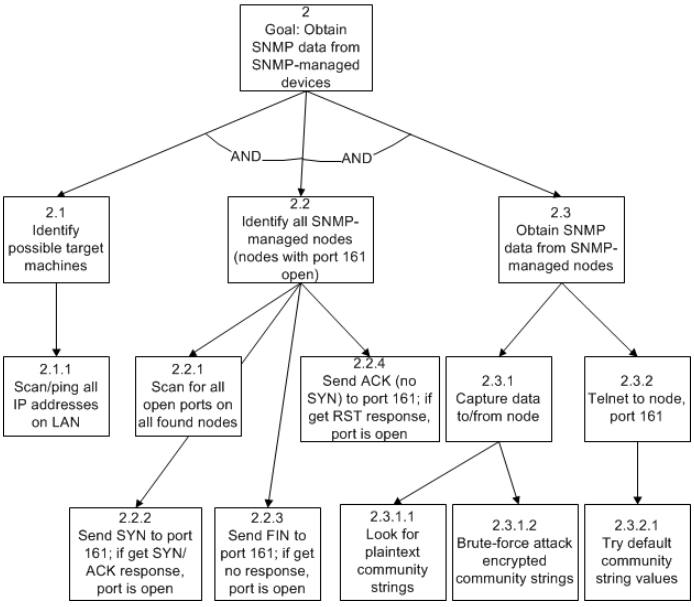

Figure 2: An example attack tree.

steps in the attack tree to its corresponding generic simple attack, if such a mapping exists. For instance, several different complex attacks include a step to identify an open port on a host. This port-scanning node in each specific complex attack was mapped to the generic simple attack of "identify an open port". When looking for the occurrence of an attack, only the generic simple attack must be identified. The mapping was then consulted to determine all the nodes in all the complex attacks that were to be colored (selected) based on the fact that the generic simple attack was attempted.

3.2 Coloring Scheme

The alerts and captured data packets were processed and attack patterns identified. If the attack was a generic attack, all

corresponding attack tree nodes were identified. The identified nodes in the attack tree were colored according to a three-color scheme: 1) green to indicate no attack occurred, 2) yellow to indicate an attack may have occurred, and 3) red to indicate that an attack most likely occurred.

Nodes are colored according to a priority assigned to the attack element. This priority is determined based on three categories of analysis: rate, access level and alert priority. These categories were chosen as input to the coloring scheme from empirical evidence of manual analysis by network managers. Manual analysis has typically been devoted to attack elements that occurred more frequently, affected a higher-level of user, or were assigned a higher priority by an IDS.

The rate is how often the element occurred in a given temporal frame and can be assigned a value of 1 through 4. If the element occurred once, the rate was assigned a value of 1, occurring more than once but less than a specified threshold is a value of 2, more than the threshold but less than twice the threshold is a value of 3 and more than twice the threshold is a value of 4. The determination of the most effective threshold is an ongoing evaluation process.

The access level is the user or privilege level gained by a successful attack. The level may vary based on the node type, such as host (a user level) or network device (privilege level). Four access levels have been defined. The lowest level is the access level, which is similar to anonymous, and gives a remote user access to a network resource. An example of this would be the web user on a host running a web server for accesses. This category is assigned a value of 0 because it is an access level provided to any remote user for specific services in a network. The user level, assigned a value of 1, is a typical user on a host. The admin (value of 2) and root (value of 3) users are separated as different access levels since they can have separate privileges based on the operating system. For instance, on a host running Windows, the administrator user is not the same as a root user on Unix since some privileges in Windows require the local administrator. For a network device running SNMP there are also SNMP privileges that can give an attacker access to device information. If the attacker gains read-only access to SNMP on the device, the access level is given a value of 1 as this is similar to a basic user on a host. Read-write access in SNMP gives the user full control of all SNMP data, including the ability to modify the device configuration, so a value of 3 is assigned to this type of attack element.

The last category considered in the coloring scheme is the snort alert priority. Snort has priorities assigned to some of the alerts raised with default values ranging from 0 to 3, with 0 being the highest priority. Since this work uses 0 as the lowest priority, this value is assigned by the formula

$$value = 3 - snort\_priority + 1 \quad (1)$$

The highest numeric value assigned in a snort priority is 3, so the snort priority is subtracted from this value to reverse the order of the snort priority (make the highest priority having a value of 0 now be the highest priority with a value of 3). One is added to this value because since the alert was raised by snort, it must be of some importance and should be considered as an attack element. Table 2 shows the three categories used to determine the attack element priority with their corresponding values.

TABLE 2
ATTACK ELEMENT PRIORITY

Category or item	Value
Rate	
Occurs once	1
1 < occurrence < threshold	2
Threshold < occurrence < 2 * threshold	3
Occurs > 2 * threshold	4
Access level	
Access (anonymous)	0
User and SNMP read-only	1
Admin	2
Root and SNMP read-write	3
Snort priority	3 – snort priority + 1

The values of the three categories (rate, access level and snort priority) are added together to obtain the priority of the attack element. The attack element, which corresponds to a node in the attack tree, is then colored accordingly. The attack element priority can be a value from 0 to 11. This range is divided evenly to determine the appropriate color for the node coloring. If the priority is less than or equal to 3, the node is colored green, if it is more than 3 but less than 8, the node is colored yellow, and if it is 8 or more, the node is colored red.

After the node is colored properly, the coloring must propagate up the attack tree. The parent nodes in the attack tree are colored based on their children nodes' colors. Nodes in an attack tree can have an OR condition or an AND condition. If the children nodes are an OR condition, then the parent node is colored with the "largest" color (green < yellow < red) of its children. Coloring the parent nodes gets more complicated if the children nodes have an AND condition. In this case, unless all the children nodes are green, in which case the parent node is colored green, the green children nodes are excluded in the color analysis. First, the majority color of the children nodes is found; if there is the same number of yellow and red children nodes, then the color of the node that was just colored is used. If the parent node is the same color as the majority of its children, then the parent node remains that color. If the parent is currently colored a higher color (red), but it was colored red more than a time threshold ago (meaning the attack was too long ago to be meaningful now), then it is colored the same color as the majority of its children; otherwise the color of the parent node is left unchanged. If the root node of the attack tree is colored yellow or red, an alert is generated that a complex attack may have occurred (if the root node was yellow) or most likely did occur (if the root node was red). The coloring scheme algorithm is outlined in Fig. 3.

For example, the attack tree in Fig. 2, may be colored based on simple attacks observed. If it was observed that the attack scanned all hosts on the network and did a telnet to the SNMP

OR conditions between children

- Color parent the color of the child with the "largest" color

AND conditions between children

- If all children are green → color parent green
- Else (skip all green children)
  - Find the color of the majority of the children (if the same number of yellow and red, then use color of latest child colored) →currColor
  - If parent color <= currColor→ color parent currColor
    Else if parent colored more than "time ago" → color parent currColor
    Else leave parent as-is

Figure 3: Coloring scheme.

port of a SNMP-managed node, these nodes are colored yellow (assuming the algorithm identified them as yellow and not red) (see Fig. 4; yellow is colored light grey in the figure and red is dark grey). The program found several ports scans over a short period of time so it was determined that this simple attack most likely occurred and the node was colored red.

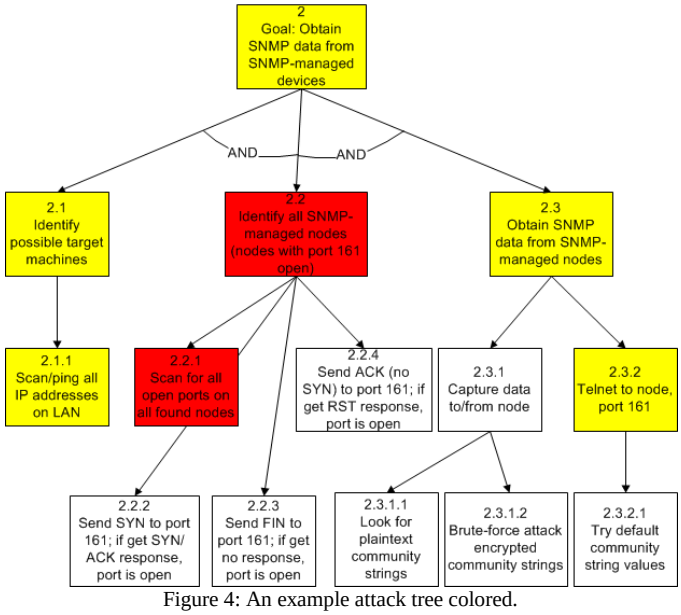

Figure 4: An example attack tree colored.

4 Simulation

Six complex attacks, varying in nature, were analyzed and tested. The first attack was an exploit of a Cisco HTTP vulnerability attack. In this attack the attacker will find a Cisco router on the network and exploit an HTTP vulnerability that will allow the attacker to download and modify the router configuration. Another complex attack, also against a router on the network, was an SNMP exploit that would allow an attacker to get access to the router's configuration file by obtaining the SNMP community strings. A TCP SYN flood attack against a router would potentially take the router off-line by consuming all the available TCP connections in the router thus causing future TCP connection requests to be denied. The common man-in-the-middle attack was also used as an example, possibly leading to eavesdropping or packet interception/modification. An RPC Locator attack was another example, which may cause 100% CPU utilization on a node causing a denial of service by the device. The last example used was exploiting a vulnerability in apache, taking advantage of chunk encoding, that would allow arbitrary code to be executed on the apache web server.

4.1 Environment

Each attack was executed in a test environment, consisting of two laptops, one running Windows XP and the other Fedora 13, a Nortel switch, and a Cisco catalyst router. The execution of each attack consisted of at least one attack path with some attacks having multiple paths attempted. Wireshark was used to generate a pcap file during each attack. A truth file was created for each attack generation / pcap file combination.

Snort and tcpdump were executed with the pcap file for each complex attack. The analysis of the snort alert file and tcpdump file was simulated in a C program. The program (see Fig. 5) created an output file for all simple attacks identified, indicating the generic attack id or specific tree node id and the appropriate color for each corresponding node in the attack trees.

1. Process the capture file (the ASCII pcap file, output from tcpdump)
   a. Read the contents of the capture file
   b. Create instances for each packet
2. Process the alert file (output from snort)
   a. Read the contents of the alert file
   b. Create instances for each alert
3. Process the packet instances
   a. For each packet instance
      For each pattern
      If the pattern exists in instance
      Write to output file for node coloring
4. Process the alert instances
   a. For each alert instance
      For each pattern
      If the pattern exists in instance
      Write to output file for node coloring

Figure 5: Steps in the simulation program.

4.2 Results

The format of the output included three fields: the date and time of the packet, the attack tree node to be colored, and the color (1 – green, 2 – yellow, 3- red). An example output for a run against the SNMP exploit example attack is provided in Fig. 6. This output is simply a textual-based version representing the nodes to be colored in the attack trees based on the simple attacks identified. By looking at the sample output, it indicates that the nodes numbered 2.2.2, 5.2.2, and 6.2.2 were all colored red (represented by the 3 in the third field of the output). The coloring scheme was then applied to color the nodes higher in the attack trees appropriately.

An attack path in each of the six complex attacks was followed and the corresponding network data captured. Fig. 4

```
08/12-16:17:51.307640  2.2.2  3
08/12-16:17:51.307640  5.2.2  3
08/12-16:17:51.307640  6.2.2  3
08/12-16:17:51.307793  2.2.2  3
08/12-16:17:51.307793  5.2.2  3
08/12-16:17:51.307793  6.2.2  3
```

Figure 6: An example output

shows an example of the colored attack tree for one of the complex attacks, after the coloring was propagated up the attack tree. The simulation was run against each complex attack. In each case, the correct patterns for simple attacks were recognized in the network data. All the correct nodes were identified and colored appropriately in the attack trees, proving the developed approach and algorithm worked as expected.

5 Conclusions

Networks are attacked daily. It is imperative that these attacks be identified, and if possible, a remedy determined. In an ideal situation, a remedy would also help prevent future similar attacks. The use of IDSs is one method used to identify attacks. Incorporating more advanced reasoning into an IDS would prove beneficial, particularly if it would lead to the remedy.

We proposed an approach to intrusion detection that used advanced reasoning through representing knowledge about attacks in ontology. Ontology will be utilized to correlate the various attack elements in a complex attack for easier identification of complex attacks. It will also allow high-level attacks to be abstracted from the low-level attacks.

Rules will be used in conjunction with the ontology to create an instance of an attack tree. Nodes of the attack tree are colored using a three-color scheme to indicate when an attack possibly occurred or most likely occurred. The root node color is used to indicate the possibility that a complex attack occurred.

The ontologies will be developed and incorporated into the system. The goal of the research is to design and implement a reasoning system using ontologies that will identify the occurrence of a complex attack as well as a remedy for the attack. The ontological representation of complex attacks will allow a better understanding of complex attacks and assist with the creation of an advanced and reusable representation of network attacks.

6 References

[1] A. Fuchsberger, "Intrusion Detection Systems and Intrusion Prevention Systems", *Information Security Tech. Report*. vol. 10, issue 3, pp. 134-139, Jan. 2005.

[2] K. Sailesh, “Survey of Current Network Intrusion Detection Techniques”, Available at http://www.cs.wustl.edu/~jain/cse571-07/ftp/ids.pdf, 2007.

[3] P. Spyns, R. Meersman, and M. Jarrar, "Data modeling versus Ontology engineering", *ACM SIGMOD Record*, vol. 31, issue 4, pp. 12-17, Dec. 2002.

[4] T. Lappas and K. Pelechrinis, “Data Mining Techniques for (Network) Intrusion Detection Systems”, Department of Computer Science and Engineering UC Riverside, Riverside CA 92521, 2007.

[5] N. Bashah, I. B. Shanmugam, and A. M. Ahmed, “Hybrid Intelligent Intrusion Detection System”, Paper presented at the *World Academy of Science, Engineering and Technology*, June 2005.

[6] M. Dass, “ LIDS: A Learning Intrusion Detection System (Thesis), University of Georgia, Athens, GA, 2003.

[7] X. Ou, S. Govindavajhala, A. Appel, “MulVAL: A logic-based network security analyzer”, *14th USENIX Security Symposium*, Baltimore, MD, Aug. 2005.

[8] M.-Y. Huang and T. M. Wicks, “A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis”, *Second International Workshop on Recent Advances in Intrusion Detection, RAID’98*, Louvain-la-Neuve, Belgium, Sept. 1998.

[9] S. A. Camtepe and B. Yener, “Modeling and detection of complex attacks”, *Third International Conference on Security and Privacy in Communications Networks and the Workshops, 2007 (SecureComm 2007)*, Nice, France, pp. 234-243, Sept. 2007.

[10] I. Gregorio-deSouza, V. H. Berk, et al, “Detection of complex cyber attacks”, *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, May 2006.

[11] S. Mandujano, “ An Ontology-supported Outbound Intrusion Detection System”, *Proceedings of the 10th Conference on Artificial Intelligence and Applications, Taiwanese Association for Artificial Intelligence (TAAI 2005)*, Kaohsiung, Taiwan, Dec. 2005.

[12] S. Mandujano, A. Galvan, and J. A. Nolazco, “An ontology-based multiagent approach to outbound intrusion detection”, *ACS/IEEE 2005 International Conference on Computer Systems and Applications (AICCSA'05)*, Cairo, Egypt, Jan. 2005.

[13] N. Cuppens-Boulahia, F. Cuppens, J. E. López de Vergara, E. Vázquez, J. Guerra, and H. Debar. “An ontology-based approach to react to network attacks", *International Journal of Information and Computer Security*, vol. 3, issue 3/4, pp. 280-305, Jan. 2009.

[14] J. Undercoffer, A. Joshi, and J. Pinkston, “Modeling computer attacks: an ontology for intrusion detection”, In G. Vigna, E. Jonsson, and C. Kruegel (Ed.), *The Sixth International Symposium on Recent Advances in Intrusion Detection*, pp.113-135, Springer, 2003.

[15] “Wireshark.” Retrieved May 7, 2010, from http://www.wireshark.org/.

[16] "Snort." Retrieved May 18, 2010, from http://www.snort.org/.

[17] B. Caswell, J. Beale, and A. Baker, Snort IDS and IPS toolkit. Burlington, MA: Syngress Publishing, Inc., 2007.

[18] “Tcpdump.” Retrieved June 10, 2010, from http://www.tcpdump.org/.