Proceedings of the IASTED International Conference
Communication, Network, and Information Security
November 14-16, 2005, Phoenix, AZ, USA

DISCOVERING ATTACK STRUCTURES USING BEHAVIOR DRIVEN ALERT CORRELATION WITH DYNAMIC VISUALIZATION OF NETWORK INTRUSIONS

Aurangzieb Zieb Rana, Mao Lin Huang, and Tom Hintz
Faculty of Information Technology
University of Technology Sydney
Broadway 2007, Australia
Email: {azrana,maolin,hintz}@it.uts.edu.au

ABSTRACT
The existing Intrusion Detection Systems often generate alerts that represent only a sub attack of the attack, which the attacker is trying to accomplish. There is no work previously been carried out to implicitly link alerts together to discover attack plans from generated alerts. This paper proposes the system frame-work for behavior driven dynamic visual intrusion detection system that can be used to find out implicit relationships among alerts and discover attack plans which consist of smaller attacks, carried out in some particular sequential order. The paper also talks about how dynamic visualization along with the static visualization can be used for the visualization of alert and attack structures.

KEY WORDS
Information visualization, Dynamic visualization, Static visualization, Intrusion detection systems.

1. Introduction

There has been significant increase over the past few years in the quantity of information. This has created a need for large scale networks and vice versa. The continued growth of e-commerce and the increase in global interconnection of computer systems has led to an increase in the number of malicious network attacks. As a result in today's wired and wireless world, attackers can cause damage worth million of dollars. The attacks can range from denial of service attack, and cyber-graffiti to corporate espionage. These attacks carried out by hackers and their tools can cause companies and individuals to lose money from loss of productivity to actual data being stolen from their machines. According to a recent survey, [3] many companies are worried about the increased sophistication of threats against their computer systems. The cause of these attacks includes poorly written software, lack of unified security infrastructure, and understaffed IT departments.

The intrusion detection systems are the tools that the organizations usually employ to protect their networks from these attacks. These tools collect network information from various advantages within the network, and analyze the information for intrusions. However there are no perfect intrusion detection systems or mechanisms. It is impossible for the intrusion detection systems to get all the packets in the network system. Especially the unknown attacks can hardly be found. The best that can be done is to optimize the way intrusion information is displayed and related to determine past vulnerabilities. The study of these past vulnerabilities then can be used to minimize the number of attacks occurring in a particular network and correct systems from these vulnerabilities by using past observations.

The Contemporary Intrusion Detection systems (IDSS) generate alerts independently, though there may be logical connections between them. These logical connections between alerts can be used to relate alerts and discover attacks consisting of smaller steps, which are raised by IDSs as independent alerts. In this paper we discuss the system frame-work for behavior driven dynamic visual intrusion detection system to logically connect alerts with each other to describe the attack that may compromise of several steps. We believe that the problem of network intrusions can be further reduced to some extent by both visualizing these attacks and network traffic generated by the IDS alerts and logically linking alerts together to discover attack lifecycle. By the use of visualization, system administrators can more quickly and efficiently identify trends on their networks. The identification of these trends in a network traffic, can even allow a below average system administrator to clearly identify the problems and vulnerabilities in their network.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 introduces our approach. Section 4 outlines our framework. Section 5 discusses its advantages. Section 6 provides conclusion. Section 7 outlines future research.

2. Previous Related Work

There have been many proposed innovative techniques both in visualization and alert correlation to facilitate detection of intrusions for computer networks. However, none of these techniques combine visualization and alert correlation in a way to develop and display attack plans and attack states to assist intrusion analysis.

The recent emerging importance of intrusion detections has allowed the development of tools like NIVA [2]. It allows interactive analysis of intrusion data to detect structured attacks using static visualization. The user can explore the three dimensional space populated by intrusion data. However it does not relate the structured attacks with each other to extract meaningful insights such as attack plans, and attack lifecycle.

Harrop, and Armitage [4] discuss the use of dynamic visualization to display real-time network traffic information. They discuss a use of visual properties, such as shape, size, location, and color to represent IP address, port number, and content of the IP packets. They have developed a prototype which provides a mapping of network metrics to visual metaphors. Again their work, as with the case of NIVA, only provides visualization and analysis of IDS logs. There is no attempt to link IDS generated alerts with each other and display them using either static or dynamic visualization to uncover relationship between attacks, direction of the attacks, and the worm activity etc.

Another article by Frincke, and Erbacher [5] describes the visualization technique that they have developed for analysis of network attacks. It involves generating an event list by parsing the database. The event list is then used for visualization. The tool that they have developed uses two dimensional space to represent various different network entities such as a circle to represent nodes, distance to represent difference in IP address of the other hosts from the node in question, and different types of lines to show traffic. The color is used to represent time of the day. Their developed tool can be used to identify attacks consisting of individual steps. However the identification can be carried out on only temporally related attacks (sub attacks).

The work by Cuppens, and Miege [1] provides alert correlation that can be used to discover causal, and implication relationships among alerts. Hence discover attack scenarios, and attack plans. They relate IDS generated alerts using full or partial matches of pre-condition for a particular alert with a post condition of some previously generated alert. The pre condition refers to the condition that is necessary for a particular attack or alert to succeed. The post condition indicates the consequence of an alert if it succeeds. These pre and post conditions are given as predicates derived from attack database described in attack specification language LAMBDA [1]. The implication relationship is discovered by using ontology rules. The paper by Cuppens, and Miege [1], concentrates on merging alerts together that refers to the occurrence of same type of an attack to reduce the number of alerts generated by IDSs. And hence provide easy management of these alerts by the security administrators. It does not deal with visualization of these alerts.

Our work deals with both the visualization of intrusion alerts as well as correlation of these alerts to achieve visualization. We combine both the information visualization especially dynamic visualization, and alert correlation to discover attacks consisting of several smaller steps. However we use our own novel techniques to both display and correlate intrusion data and find the implicit relationships between attacks. They will be discussed in the next section of this paper.

The work presented by Cuppens, and Miege [1], is closest to ours in respect to alert correlation. However our approach has several differences to suit our purpose. Whereas the purpose of their work is to merge similar alerts together. Our aim is to investigate attacks and attack steps. Our method allows alert aggregation both during and after alert correlations, while their approach only treats alert as a separate stage before the correlation of alerts. By allowing alert aggregation both during and after alert correlations, our technique can allow interactive analysis of attacks consisting of smaller sub-attacks. Our work also uses visualization to display state information of a particular attack. For instance the attack can be at a growing stage or at a maturing stage when it has affected the targeted machine/s. The state information of an attack will allow the security analyst to visually discover life cycle of a particular attack and hence this will increase their understanding about the structure of the attack.

3. Proposed New Approach

In this section we discuss our technique used for alert correlation to discover attack plans. The approach consists of two steps. The first step consists of grouping alerts such that alerts within a single group have similar characteristics, while alerts in different groups are dissimilar without any similarties. The second step involves relating implicitly alerts together within the same group with each other using predicates to discover attacks consisting of smaller IDS generated alerts.

3.1 Grouping Alerts

The alerts can be grouped together or formed into clusters based on some similarity rules. We define our similarity rules in the form of concept hierarchies [6]. The concept hierarchy is an important source for inductive learning [6]. We use concept hierarchies for attributes of the generated alerts to derive similarity between alerts. Figure 1 on the next page shows how the concept hierarchy can be used for both the IP address and port number. The concept hierarchy will provide a generalization of hierarchy on selected attributes of the alerts. For instance a concept hierarchy on a particular attribute such as port number may give port 80 as a private port. The two alerts can then be related by closer their attributes such as port numbers are using the concept hierarchy for a port number. For example if port numbers from both alerts get generalized to private port then they can be related to each other by this generalization.

The IDS generated alerts consists of several attributes such as source ip address, destination address, and alert type. These attributes can be used to define the alerts as tuples over the Cartesian product $Z_{a1} \times Z_{a2} \times \ldots Z_{an}$ where Z_{an} represents the possible values for some a_n attribute of some alert. We can use this technique of alert tuples over the Cartesian product to define a *concept hierarchy* for any particular attribute. The concept hierarchy will be in the form of a tree on all the elements of Z_{an}. For two elements y, $\hat{y} \epsilon Z_{an}$. The $\hat{y}$ can be called a parent or generalization of y if there is a direct edge from $\hat{y} \rightarrow y$ in the concept hierarchy for Z_{an}.

Figure 1 shows the concept hierarchies for both an IP address and port number. The diagram illustrates how the concept hierarchies can be used to derive the similarity among alert attributes. From Figure1a we can see Ip address Ip7 is a www-server, is a internal machine, is any Ip address. We now proceed to show more clearly this relationship. The relationship can be expressed as Ip7 $\leftarrow$ WWW-server $\leftarrow$ Internal $\leftarrow$ All-Ip. Similarly Ip address Ip8 is a www-server, is an internal machine, is any Ip address. This can be expressed by Ip8 $\leftarrow$ WWW-server $\leftarrow$ Internal $\leftarrow$ All-Ip. The two IP addresses Ip7, and Ip8 can be related by the generalization of their values using the concept hierarchy as shown in the diagram. Both have www-server as their parent, which in turn has internal machines as its parent and so on. We can relate them by using www-server as the first level generalization.

So far we have shown the use of Ip addresses and port numbers to achieve similarity relations among alert attributes. We can also include time-stamps. The concept hierarchy for the attribute timestamp can be produced to generalize the timestamps based on their time of occurrence. For instance the time-stamp t_1 can be a Monday, which is a week day, and finally week day can be any day of the week.

We can extend this approach from comparison of attributes according to their similarities using concept hierarchies to alerts in general. By using a similar notation as we have used above, let A, and $\grave{A} \epsilon (Z_{a1} \ldots Z_{an})$ be two alerts. By again referring to Figure1 clearly we can conclude alert À is a parent of A if there is an edge A $\leftarrow$ À and also for all attributes of A and all attributes of À there are edges such that for each attributes $\grave{a}_i \epsilon \grave{A}$ there is a directed edge to $a_i \epsilon A$. If these conditions hold we can generalize the alerts to specific groups or clusters. Since as can be seen from the above paragraph the similarity depends on the level of generalization. Next we define the level of generalization as $\beta(\grave{A},A) = \sum \beta(\grave{a}_i \epsilon \grave{A}, a_i \epsilon A)$ for all i=1 to i=n. We need to achieve minimized level of generalization to achieve meaningful similar alert clusters. For instance we can group alerts together using a generalized alert represented by a triple, *(FIREWALL, PRIVATE PORT, MONDAY)*. This will allow any alerts that have their destination IP addresses belonging to a firewall category in a concept hierarchy for all Ips with ports of communication belonging to a private port in a concept hierarchy shown for ports in the diagram below. And time stamps which can be generalized to Monday using the concept hierarchies for time stamps to be grouped together for further investigation.

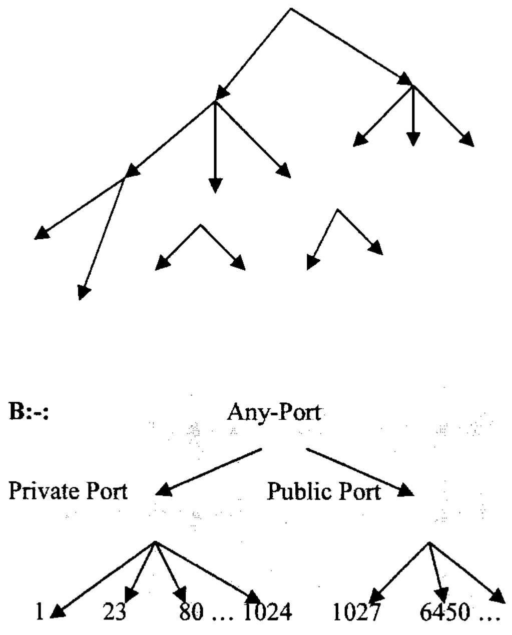


Figure1. The figures both A and B shows concept hierarchy for the source or destination IP address and port number of an alert.

The next section discusses how the group of similar alerts derived by using above approach can be further related to each other using predicates to discover attacks consisting of several smaller IDS generated alerts.

3.2 Correlating Alerts

This section introduces an approach that can be used to implicitly link alerts together with each other based on using predicates and ontological rules. The alerts may have associated with them pre-conditions and post conditions to specify requirements for an alert to succeed and effects of an alert [1]. These pre, and post conditions can be represented by the set of predicates. These predicates can then be used to relate alerts together based on a common relationship among predicates. To further improve our correlation model for alerts, we introduce a set of ontological rules containing pair of pre and post conditions for attacks. These rules can then be used to indirectly link alerts together to discover implicit links between them. By implicitly linking alerts together we can discover links between earlier attacks that are used by attackers to carry out later attacks. The next two sections discuss these techniques in more detail.

3.2.1 Using Predicates. Clearly if an early attack is to prepare for a later attack, the post condition of the earlier attack should at least partly or completely satisfy the pre condition of the later attack. We can use predicates to represent pre/post conditions of attacks. The predicates can then be used to discover the similarity relations among alerts. For instance a port scanning alert generated by particular IDS may discover TCP service vulnerable to a certain buffer overflow attack. We can use a predicate *TCPVULTOBOF (Ip_victim, Port_victim)* to represent the post condition (effects) of an attack. Now if another attack that is detected and reported by an IDS requires the TCP service sensitive to the buffer overflow attack for the same victim Ip and port, then we can use the same predicate to represent a pre-condition for an alert generated for this attack. And both alerts can be correlated with each other by the use of same predicate.
Next we introduce a notion of generic alert types, which will have associated with them pre/post conditions for attacks. They will be used to represent the pre and post condition of each type of alert and encode the knowledge about a type of attack. The instances of different generic alert types will be used to represent alerts received from IDSs. Each *generic-alert type* G will consist of a triple (*set(attributes)*, *pre-condition*, *post-condition*), in which *set(attributes)* is a set of attribute names, and *pre-condition* or *post-condition* is a logical combination of predicates whose variables are all in *set(attributes)*.

For an instance *h* of *Generic-alert type* G, if the logical combination of all its *pre-conditions* is true, then it implies that this attack is successful. And if logical combination of *post-conditions* is true, then that implies the possible consequent attack. The instance *h* will consist of a finite set of tuples $\in$ *set(attributes)* of *G* with each tuple having associated interval based timestamp (begin, and end time). When given alert instance *h of Generic-alert type G,* the *pre-condition of h* (or *post-condition of h*) will be the predicates of *G* indicating pre/post conditions for *G*. We may have conjunction or disjunction of predicates to represent pre and post conditions for a particular alert. Hence we will have the set of all predicates that appear in both pre and post conditions of *G*, denoted by *Pre(G)*, and *Pos(G)*. Similarly for an instance *h* they will be represented as *Pre(h),* and *Pos(h).* Now to relate two instances h1, and h2 together, we can do the following, if there exists *pred* $\in$ *Pre(h2)* and *(pred)* $\subseteq$ *Pos(h1)* such that for all *p* $\in$ *(pred)*, p.end-time < pred.begin-time and the conjunction of all the predicates in *Pos(h1)* implies *pred* $\in$ *Pre(h2)* then by this approach, we can capture the implicit relationships between these two alert instances. By saying that occurrence of instance h1 has caused an occurrence of h2. The implication here also means having common *predicates pred* $\in$ *{Pre(h2), Pos(h1)}*. And relating instances based on these common predicates.

This technique will also allow us to reduce the number of false positive alerts. For instance suppose there is a sequence *Seq* of generic-alert instances, an instance *A* $\in$ *Seq* is a *correlated instance* if there exists another $\acute{A}$ such that either *A* prepares for $\acute{A}$ or $\acute{A}$ prepares for *A*. If *A* is not correlated with other instances, we can call this alert *A* as a one-off event or a false positive. And can remove it from the sequence.

We can use directed acyclic graphs to represent relationships among alert instances. They allow us to create precedences among alerts. We can use alert instances created from generic alert types as nodes in the directed acyclic graph and edges to create precedence among instances. For instance for some sequence *Seq={A1, A2, A3,A4,A5}* of alert instances by using our alert correlation approach as stated above, assume we discover the *Pos(A1)* implies *Pre(A2)* » *Pos(A2)* implies *Pre(A4)* » *Pos(A4)* implies *Pre(A5)* » *Pos(A5)* implies *Pre(A3)* » *Pos(A3)*. This finding will allow a discovery of the actual attack and its lifecycle consisting of smaller steps in the form of the alert instances. The attack can then be represented as directed acyclic graph as shown below.

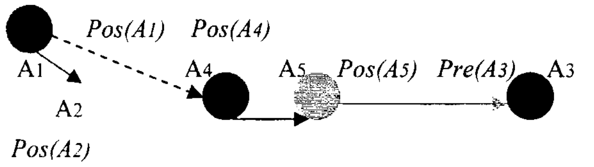


Figure2. The figure shows directed acyclic graph for an attack consisting of several smaller steps. The broken arrow indicates partial implication of Pos(A1) and Pre(A4).

3.2.2 Using Ontological Rules. Next we discuss the use of ontologies as a high level conceptual modeling approach for describing knowledge that can be used to implicitly link unrelated alerts together by the use of ontological rules. The work presented by Cuppens, and Miege [1], uses ontological rules to relate alerts. However they use ontologies to only merge similar alerts together and do not use them to discover attack steps as in our case. We use ontologies to describe implication relationship among predicates.

Suppose we have two alert instances A and À without any common predicates between them. Clearly we cannot relate these two instances together by using *pre(A)*, and *post(À)* without having any implication relationship between *Pre(A)* and *post(À)*. We can derive this implication by using ontological rules. By using ontological rules we can find an indirect link between alerts. The ontological rules will be in the form of pair of pre and post condition predicates outlining a necessary condition for an attack. For instance suppose as in [1] we have an alert instance which involves port scan and another instance which involves carrying out a denial of service attack against a window based OS machine. Now in order for a denial of service attack to succeed the attacked machine needs to be windows based. The attacker can carry out port scan for scanning a NetBios port to check whether the NetBios session is open to carry out their attack. The port scan alert instance A may have an associated with itself a post-condition predicate *ExistHost(ip=129.122.12.1, port=139 (NetBios))*. Next we may have another generated instance À for denial of service attack for windows based machine with a pre-condition predicate stating the pre-condition for an attack as *Os(ip=129.122.12.1, Os_type=Windows)*. Both A, and À will not be related with each other. However by having an ontological rule say R, we can state both predicates as pre, and post conditions for an attack in their general from as *ExistHost(ip_address, port=139 (NetBios))*, and *Os(ip_address, Os_type=Windows)* and relate them by using the R with each other. The figure given below generalizes this idea and illustrates how ontological rules can be used to discover indirect connections between alerts. The rules can be related with each other and alerts with rules to find indirect links between alerts.

Once an attack is discovered by using above techniques we can further analyze the attack by concentrating on a particular aspect of an attack such as destination ip address and related attacks with common aspect. We can use focusing constraint that will also be a predicate, and relate attacks for which the focusing constraint is true such as Known(ip_source,ip_dest).

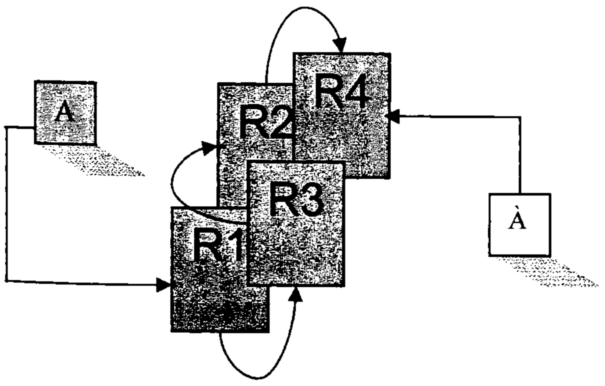


Figure3. The figure shows how ontological rules can be used to indirectly relate alerts together, the alert A is related to alert À via R1, R3, R2, and R4.

4. System Frame Work

In this section we present the frame work of our proposed system (see Figure4) and discuss the design and development of our on-line visual intrusion alert correlator.

The alert preprocessor initially processes the raw alerts to transform them into a unique format for easy analysis. They are then stored into a relational database *alert Db1*. The cluster generator is used next to group alerts together based on similarity rules according to the conceptual hierarchies explained in section 3.1 of this paper. The resulted groups are stored in a another relational database *Db2* for further processing. The visualization console is connected to *Db2* for clustered visualization of alert data. The alert instance generator uses the *Db2* to generate instances of alerts based on generic alert types with pre/post conditions. It will generate one instance of a particular generic alert type from each alert and aggregate multiple similar alerts into one alert instance using the information in the expert knowledge database. The expert knowledge database will contain the necessary information about generic-alert types as well as implication relationships between predicates in the form of ontologies. We can store these relationships between predicates in a XML file. The diagram given on the next page shows the system architecture for our proposed system along with all the required components for the system.

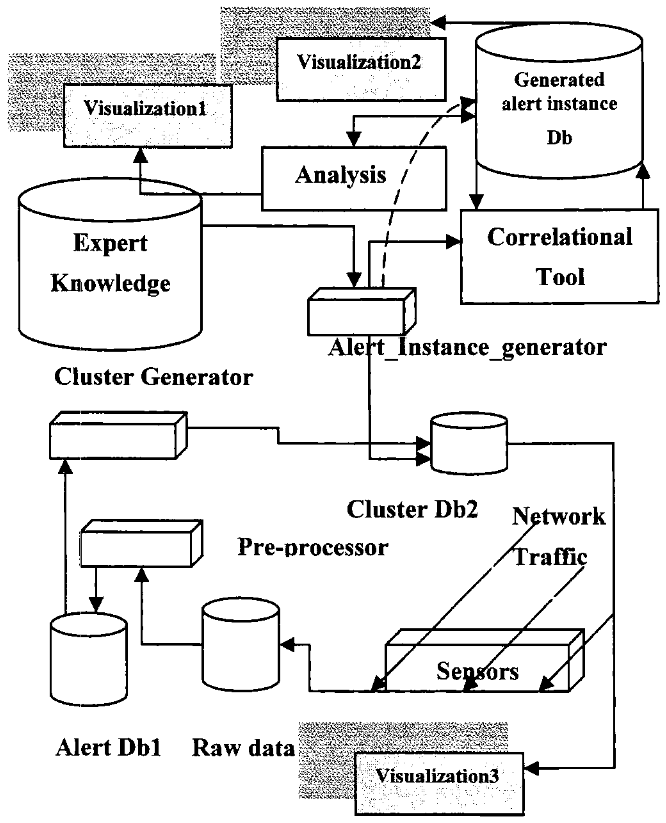


Figure4. The figure shows the system architecture of our proposed system with all the necessary components.

The correlation tool is used for performing the actual correlation task using the alert instances for various groups of alerts created by the instance generator and the auxiliary data provided by the expert knowledge database. When the correlation tool reads in the alert instances, it generates the pre/post condition sets of each instance and uses them to find all the prepare-for relationships between generic-alerts types instantiated by the alert instance generator. After alert correlation phase, the correlated alerts are inserted into a *generated alert instance Db* for analysis. The correlated alerts from the database are extracted and are transformed into readable form (graphs, text, and glyphs) to user. This operation is done by an analysis tool, and visualization consoles as shown in the Figure 4. Next we discuss in greater detail the visualization for our proposed system.

4.1 Visualization

This section of the paper discusses the design of our information visualization tool to support viewing, monitoring, and analysis of correlated alerts derived by using methods described in section 3. The computer security specialists use output from IDSs along with other resources such as network and firewall logs to keep abreast of potential security threats. These outputs employ tools such as NIVA [2] for visual displays of alert data to reduce false positives and reduce cognitive load of the analyst by taking advantage of information visualization to amplify cognition [8]. Although information visualization is a natural choice for displaying intrusion alerts. However there has been little research into implicitly linking alerts together and displaying them as part of the same attack. Our visualization technique is based on displaying alert data using 2D and 3D displays. Information visualization principles described by Shneiderman [9]: "overview first, zoom and filter, then details on demand" will be used in our visualization approach. We employ three types of visualization techniques integrated together by a single visualization console. The clustered visualization will be used to visualize grouped alerts according to similar properties. The database visualization will be used to display explicit relationships among alerts using two and three dimensional displays and by using static visualization. The dynamic visualization will be used to display attacks and their lifecycles. The attack lifecycle will be formed by discovering implicit relationships among alerts.

4.1.1 Clustered Visualization. Our visualization tool will use multi level clustering to display IDS generated alert information. The clustered visualization will follow Shneiderman's [9] visualization principles compromising of overview and zooming of the collected alert data. The clusters will be formed from group of attackers performing the same kind of activities on the home network machines. These clusters will be used to generate second level clusters showing external computers attacking more than one home network computer distinguished by attack types. The second level clusters will be used for obtaining a more detailed view consisting of home network hosts affected by a particular external host belonging to a particular level two cluster. Hence second level clusters will be used to obtain third level clusters consisting of home network hosts according to attack type by an external host machine. The third level cluster formation will allow security analyst to see which local hosts are affected by which type of an attack.
The third level clustering will be used for obtaining an overview of the collected alert data. Now to extract deeper insight into the gathered data, we will allow zooming. The zooming will be achieved by selecting a single or multiple hosts from a level three cluster. The zooming will allow the security analyst to analyze intrusive traffic between selected residential hosts and external hosts.

In order to facilitate zooming, a representation of residential hosts will be shown with lines from individual hosts communicating with an external node in the ip space. The lines will be used to encode attack information. The color for lines will be used to represent the directionality of the communication. For instance a red line could represent a dual communication between a computer on a home network and an outside network computer. The line will be used as an indicator to see if a residential host on home network both received and sent data packets to an external host. Similarly a blue line could represent a packet sent from a home network computer, which never received a response. A yellow line can be used to encode a packet that was sent to a home computer, but the home computer never replied back. The Figure 5 illustrates how the line encoding will actually work to discover suspicious activity. The lines are colored red.

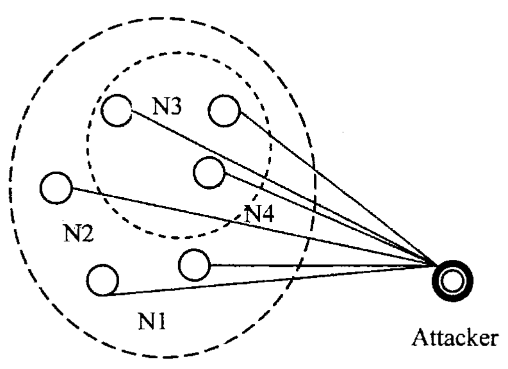


Figure5. The figure shows an example of ping sweep attack by an external host.

Figure 5 shows an instance of ping sweep attack. The attacking node is located outside the dashed circle. The filling of the outer circle of the attacking node indicates the external node status. The ring of circles indicates hosts which have been attacked. The color of the rings shows the protocols used by the attacking node. The local nodes will encode port information to indicate what ports have been opened up for communication with an external node. To accommodate changes in a communication pattern between an internal node, and an external node, a time slider will be incorporated.

4.1.2 Database Visualization. Next we discuss database visualization, which will allow both filtering and details on demand described by Shneiderman [9]. The clustered visualization will allow the visualization of the communication between internal and external hosts. The next insight that is desired is to know how many packets were communicated between external hosts and internal residential hosts. Many current systems do not include this information. It is not simply enough to show communication between internal network computers, and external hosts. There is a need to also show how much communication actually happened. This can be provided by providing an interactive visual environment compromising of two areas base plane, and floating nodes [2]. The base plane will contain three dimensional bars representing external nodes attacking internal network hosts. The size of the bars will indicate net packet exchanges with the internal hosts with taller bars indicating large amount of packet exchanges.

By clicking on each of these bars the circular floating nodes will be shown on top of the base plane. These nodes will be internal hosts affected by a particular external host. The floating nodes will be shown as spheres with size of the sphere will indicate total number of packets exchanged with a particular external node. At any time the operator of the system will be able to also see what percentage of the overall traffic a particular computer contributes to an external host. It will be shown as a label next to a sphere in floating nodes. This type of database visualization will allow a security analyst to build a cognitive map of their findings and help them detect attack patterns and hence planned structured attacks.

4.1.3 Dynamic Visualization. The 2D displays will be used for continuous monitoring without the need for focused attention. They will provide a starting point for recognizing and flagging alerts that require further analysis in a way that can be done quickly and effectively as well as visualize the evolution of an attack using graph drawing technique defined by Collberg [10]. They will provide overview information with ability to allow exploration, and filtering of data and provide details on demand as described by Shneiderman [9]. The dynamic queries [8] will be used for interactive exploration of alert data represented by 2D displays. The dynamic queries enable a very powerful mechanism for information visualization by allowing visualization to be modified on the fly as security analyst modifies the range of interest within the domains of the various attributes of the visualized information. The diagram given below gives a user interface to our 2D display.

The diagram only illustrates the basic user screen for attacks occurring on a particular host. The prototype will have a more detailed screen for displaying attack information for individual hosts including double-dragbox sliders for date, month, and year etc. The attack area on the left hand side of the user screen shows the attacks occurred or currently occurring attacks. The glyphs are colored to indicate the types of an attack and attack severity level. The glyphs shaped as diamond labeled with *?* indicates currently undetermined attacks, the attacks that are still in progress. The screen also allows user to switch to a display for individual attacks containing several steps for a particular attack and visualize attack states. This is done by clicking on an individual attack. The screen for this display can be seen on figure 7.

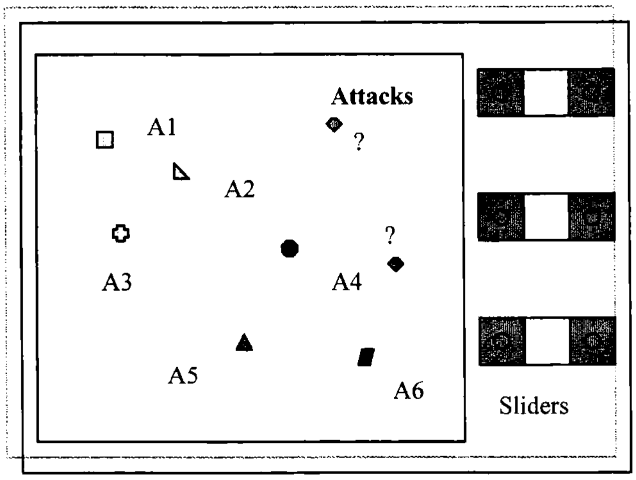


Figure6. The figure shows the basic information visualization screen for displaying network attacks for individual hosts.

The screen shown in Figure 7 will display related alerts with each other as shown above along with state information of the attack by pointing over to a particular node with a mouse. The state information will be a text based information describing consequence (post condition) of a particular alert. By allowing the post condition of each alert to be displayed, we can show the state of the attack during its different phases. The dotted line in the diagram shows indirect links between alerts, where the post-condition of one alert has partially satisfied the pre-condition of another alert.

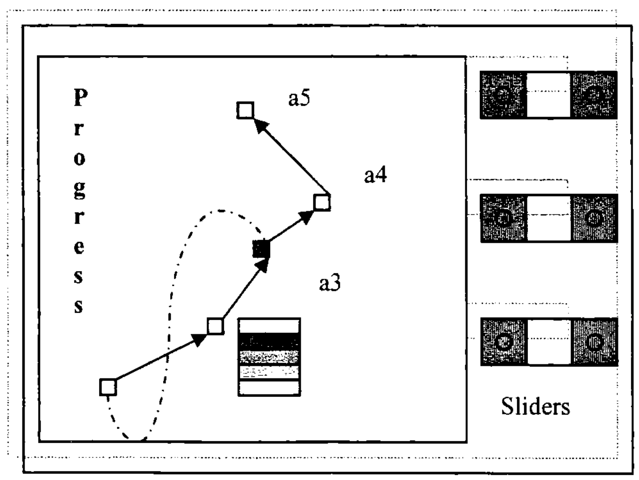


Figure7. The figure shows the basic information visualization screen of a single attack consisting of several smaller steps.

The double dragbox sliders are disabled here in this screen. The drop down menu as shown in the diagram above as a colored rectangle will allow the user to switch between different display screens, both 2-dimensional and 3-dimensional. It will allow the user to both go back to the main navigational screen or a 3-dimensional display. The option will also be provided by the drop down menu for a multiple displays illustrating attacks on a particular host related by common pre/post conditions. These conditions will be displayed on the user screen for highlighting common similarities between attacks. The multiple views will also be used for representing different stages of the same attack differentiated by time. The color of the nodes will indicate alert severity level with red for instance indicating very severe alert and light blue indicating milder alert.

5. Conclusion

We presented the frame-work for a behavior driven dynamic visualization system for network intrusions to discover attacks which consist of a series of smaller attacks preparing for later attacks. The development of such a tool will enable system administrators to more quickly and efficiently identify attack trends. The types of packets sent, which ports were used, the number of packets sent etc. The visualization console will allow the user to not only see information contained in thousands of lines in a single snapshot but also discover state information of a currently occurring attack. Our work will allow the removal of isolated alerts, the alerts that are not co-related with other alerts, and hence also reduce false positives.

6. Future Work

Our next step is to produce the initial prototype which uses our correlation model for implicitly linking alerts together and provide dynamic visualization to query the alert database for changes in the database and reflect these changes in the visualization console. Once the initial prototype is complete, we will extend it by having cross correlation between alerts belonging to different alert clusters derived by conceptual hierarchies.

References

[1] F. Cuppens, A. Miege, & O. Toulouse, Alert correlation in a Cooperative Intrusion Detection Framework. In a *proc of the 2002 IEEE Symposium on security and privacy*, Washington USA, 2002, 202-216.

[2] K. Nyarko, T. Capers, C. Scott & K. Ladeji-Osias, Network Intrusion Visualization with NIVA, an Intrusion Detection Visual Analyzer with Haptic Integeration. In *proc of the 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Orlando Florida USA, 2002, 277-284.

[3] 2003 Global Security Survey, http://www.deloitte.com/dtt/cda/doc/content/Global%20Security%20Survey%202003.pdf.

[4] W. Harrop, G. Armitage, Intuitive Real-Time Network monitoring using visually orthognal 3d

metaphors. In *proc of 2004 Australian Telecomunication Networks and Applications Conference*, Bondi Australia, 2004.

[5] R. Erabacher, & D. Frincke, Visualization in Detection of Intrusións and Misuse in Large Scale Networks. In *IEEE Internacional Conference on Information Visualization*, London England, 2000, 294-302.

[6] J. Han, Y. Cai, & N. Cercone, Data Driven Discovery of Quantative Rules in Relational Databases. In *IEEE Transactions on Knowledge and Data Engineering*, Piscataway USA, 1993, 29-40.

[7] L. Good, & B. Bederson, User interfaces as a medium for slide show presentations. Information Visualization Volume 1, Issue1, Palgrave Macmillan, USA, 2002, 35-49.

[8] B. Schneiderman, Dynamic Queries for Visual Information Seeking. IEEE Software archive, IEEE Computer Society, USA, 1994, 70-77.

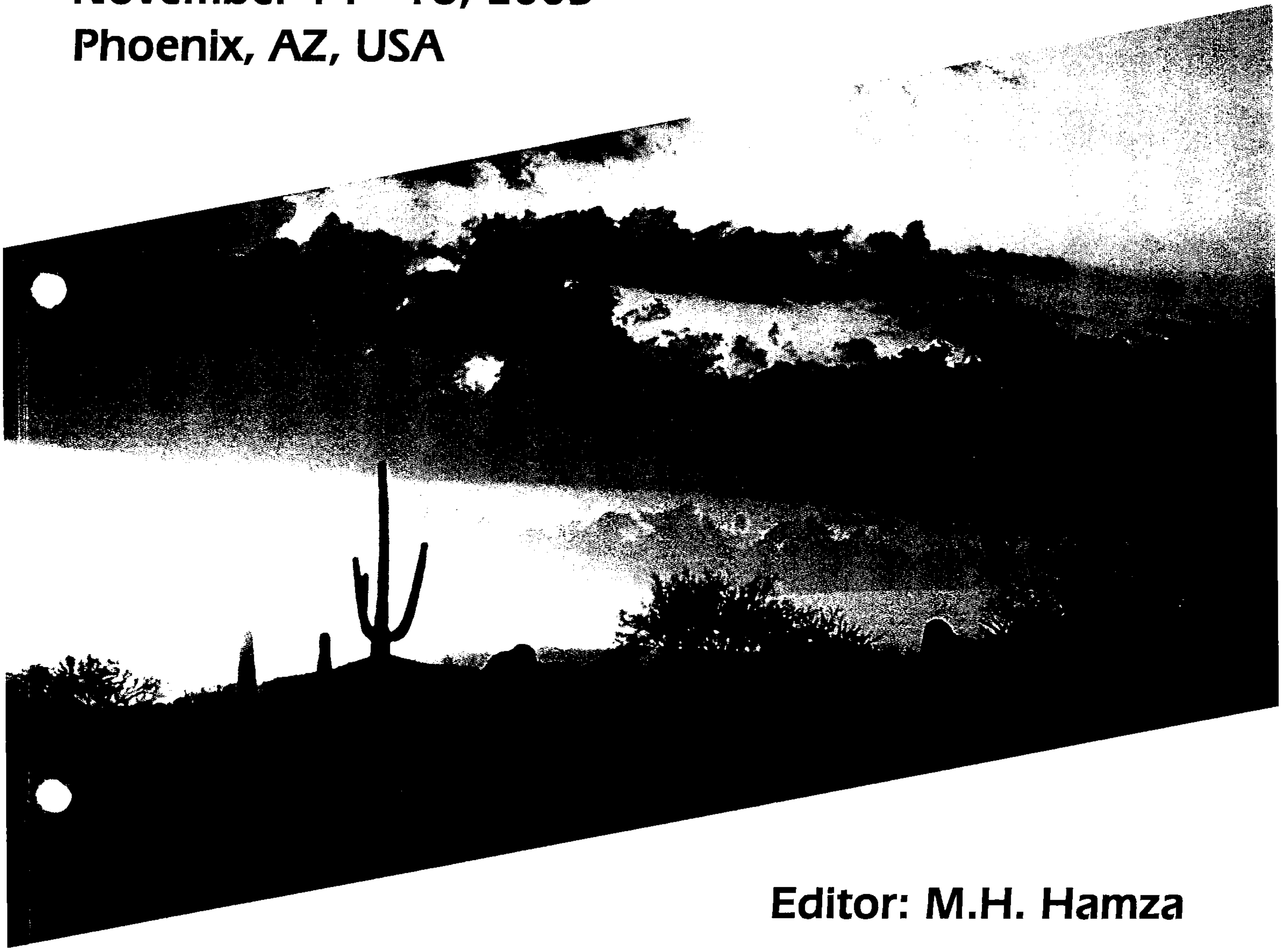
[9] B. Shneiderman, The Eyes have it: A task by data type taxonomy for information visualization. In *proc IEEE Symposium on visual languages*, Colorado USA, 1996, 336-347.

[10] C. Collberg, S. Kobourov, J. Nagra, J. Pitts, & K. Wampler, A system for graph-based visualization of the evolution of software. In *proc 2003 ACM Symposium on Software visualization*, NY USA, 2003, 77-86.

Proceedings of the IASTED International Conference on

Communication, Network, and Information Security

November 14 - 16, 2005
Phoenix, AZ, USA



Editor: M.H. Hamza

IASTED A Publication of the International Association of Science and Technology for Development

ISBN: 0-88986-537-X

ACTA Pres
Anaheim | Calgary | Zuric

Proceedings of the IASTED International Conference on **Communication, Network, and Information Security**, held November 14 - 16, 2005, Phoenix, AZ, USA.

SPONSORS

The International Association of Science and Technology for Development (IASTED)
- Technical Committee on Communication, Network, and Information Security
- Technical Committee on Information Systems
- Technical Committee on Networks

EDITOR

M.H. Hamza – IASTED, Canada

INTERNATIONAL PROGRAM COMMITTEE

ADDITIONAL REVIEWERS

For each IASTED conference, the following review process is used to ensure the highest level of academic content. Each full manuscript submission is peer reviewed by a minimum of two separate reviewers on the International Program Committee/Additional Reviewers list. The review results are then compiled. If there is a conflict in reviews, the paper is sent to a third reviewer.

ACTA Press
P.O. Box 5124
Anaheim, CA 92814–5124
USA

ACTA Press
#80, 4500 – 16 Ave. NW
Calgary, AB T3B 0M6
Canada

ACTA Press
P.O. Box 354
CH–8053 Zurich,
Switzerland

Publication Code: 499

TABLE OF CONTENTS
CNIS 2005

AUTHENTICATION

499-021: The Achilles' Heel of JPEG-based Image Authentication
M. Schlauweg, T. Palfner, D. Pröfrock, and E. Müller 1

499-042: Modelling a Biometric Authentication Protocol for 3G Mobile Systems using CSP
S.A. Shaikh and C.K. Dimitriadis 7

499-053: Two Practical Attacks against Bluetooth Security using New Enhanced Implementations of Security Analysis Tools
K.M.J. Haataja 13

499-071: Anonymous Communications on the Internet
Y. Wang and P. Dasgupta 19

499-093: Mobility Management for UMTS-WLAN Seamless Handover; within the Framework of Subscriber Authentication
H. Kwon, K. Rho, A. Park, and J. Ryou 25

499-813: A New Authentication Protocol for Revocable Anonymity in Ad-Hoc Networks
A. Wierzbicki, A. Zwierko, and Z. Kotulski 30

SPAM AND INTRUSION DETECTION

499-017: Token Buckets for Outgoing Spam Prevention
W.N. Gansterer, H. Hlavacs, M. Ilger, P. Lechner, and J. Strauß 36

499-030: Preventing Spam by Dynamically Obfuscating Email-Addresses
T. Eggendorfer and J. Keller 42

499-069: LOHIT: An Online Detection & Control System for Cellular SMS Spam
S. Dixit, S. Gupta, and C.V. Ravishankar 48

499-014: Discovering Attack Structures using Behavior Driven Alert Correlation with Dynamic Visualization of Network Intrusions
A.Z. Rana, M.L. Huang, and T. Hintz 55

499-023: Preventing Information Leaks in Xwindows
V. Diwakara, V. Rajegowda, and P. Uppuluri 64

499-063: Characterizing and Estimating Network Fluctuation for Detecting Interactive Stepping-Stone Intrusion
J.H. Yang and S.-H.S. Huang 70

499-078: A Novel Service-Oriented and User-Centric Intrusion Detection System for Ubiquitous Networks
B. Zhou, Q. Shi, and M. Merabti 76

499-801: Anomaly Detection in a Distributed Environment using Neural Networks on a Cluster
N. Srinivasan and V. Vaidehi 82

499-804: A Hierarchical Approach for Detecting System Intrusions through Event Correlation
G.W. Rice and T.E. Daniels 88

CRYPTOGRAPHIC PROTOCOLS

499-022: Efficient Hierarchical Conference Key Establishment in Wireless Networks
S. Eskeland 94

499-034: Extending Constraint Solving for Cryptographic Protocol Analysis with Non-Standard Attacker Inference Rules
S. Malladi and S.J. Rosenberg 99

499-086: A PKI-based Secure Audit Web Service
W. Xu, D. Chadwick, and S. Otenko 106

499-088: A Compact and Low-Power Implementation of the Montgomery Algorithm for a Public-Key Cryptosystem
J. Rejeb, C. Madaboosi, and T.T. Le 111

499-091: An LSB-based Watermarking Algorithm for Grayscale Images
C. Obimbo and Y. Zhang 118

499-803: Design and Implementation of a New Public-Key Certificate Status Handling Scheme
E. Faldella and M. Prandini 125