Proceedings of the 2002 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 17–19 June 2002

Closing-the-Loop: Discovery and Search in Security Visualizations

Kiran Lakkaraju, Ratna Bearavolu, Adam Slagell, William Yurcik,

Abstract—

The tasks of security engineers include detecting attacks and responding to them. In order to accomplish this, a security engineer must be able to decide what behavior indicates an attack and then search for this behavior. Current security visualization tools provide rich and concise visualizations of network data that allow security engineers to determine the nature of attacks on the network. However, current security visualizations lack the ability for security engineers to search for these behaviors in the network logs. The process of finding interesting patterns in the data is called *discovery*, and finding instances of these patterns is called *searching*. Security engineers must do both discovery and search, but current security visualization tools only help in discovery.

In this paper, we describe the modifications we have made to our security visualization tool, NVisionIP, that allow security engineers to not only discover patterns in the data, but also to search for those patterns in other data.

I. Introduction

The field of security visualization has experienced tremendous growth since its inception a few years ago. One of the main reasons for interest in this area is that automated techniques for detecting attacks are prone to false positives. Visualization tools reduce these false positives by putting humans back in-the-loop. While this growth is impressive, it has been leveling off, and we believe that security visualization is reaching a turning point. While in the future there may be new types of security relevant data to display, there are a limited number of ways to usefully visualize the information in network traces—the type of data that has been the focus of security visualization to this point. Much of the recent work in this area has been feature creep in tools or incremental improvements, rather than fundamental changes or paradigm shifts. We believe the next fundamental breakthrough will be decoupling search from discovery in what we call *closing-the-loop*.

The idea of closing-the-loop comes from the following observations. Humans are particularly skilled at discovering patterns, especially visual ones. However, they perform poorly at pattern matching within large data sets. Computers excel at the opposite operations; they lack the intelligence to discover many patterns, but given a search string, they are excellent at matching a pattern in a large data set. We should seek to exploit the strengths of both the human and the machine.

The loop in our analogy starts with log data collected by machines. Log data is then visualized and passed to a human operator for pattern discovery. The loop is then closed when this pattern is translated into a query that computers can understand and use to process the raw data again—not necessarily the same data set. Current visualizations do not close the loop and stop after presenting the data to a human operator. What we propose is separating the tasks of discovery and search so that the most apt entity can focus on the particular task; humans will discover and machines will match patterns or search.

While the types of discovery and pattern matching that would qualify as closing-the-loop are almost limitless, in this paper we focus on a specific implementation. We describe how we have taken our NetFlow visualization engine, NVisionIP, and added mechanisms that allow users to visually create search patterns. These search patterns can then be used by a second application to process the raw logs.

Section II discusses related work. In Section III we further illustrate the difference between search and discovery and argue why the two tasks should be separate. Section IV discusses how we implement the idea of closing-the-loop in the NVisionIP environment. Lastly, we conclude and discuss future work in Section V.

II. Related Work

Visual data mining (VDM) focuses on using visualization to describe the data and patterns created by data mining (DM) algorithms. The visualization allows the user to discover more patterns in the data, evaluate the current patterns and then re-run the data mining algorithms with different inputs. The goal of VDM, though, is the use of visualization to help humans create patterns or evaluate the patterns of the DM algorithm. Our work focuses on allowing the creation of patterns within a visual environment, and it thus has some links to VDM [1].

There has been much work on visualizing networks, [2] describes many of the early visualizations of the Internet. Some of the visualizations are geographical in nature, showing the traffic flow between machines as a link between the physical locations of the machines. Other visualizations

Kiran Lakkaraju: NCSA, University of Illinois, Urbana-Champaign, IL, Department of Computer Science, University of Illinois, Urbana-Champaign, IL

Ratna Bearavolu, Adam Slagell, William Yurcik:NCSA, University of Illinois, Urbana-Champaign, IL

ISBN 0-7803-9814-9/$10.00

focus on connectivity patterns and traffic volumes. [3]

In terms of visualization for security, [4] provides an example using BGP routing data. Although the data has been visualized to look for security incidents on the Internet, this work does not provide a sense of situational awareness as it analyzes traffic between autonomous systems.

A new tool to enhance situational awareness is the Spinning Cube of Potential Doom [5]. This tool represents network traffic as points in 3D space. The addresses of the network being monitored lie on one axis, all possible source IP addresses lie on a second axis, and the third axis represents port numbers. The color of the points represent different characteristics of the traffic flows on the network. This presentation is similar to that of NVisionIP, though it tends to be more "busy". Although similar to NVisionIP, the Spinning Cube of Potential Doom does not allow the user to drill down or filter for events of interest.

One of the key differences is that we explicitly allow the user to create patterns from the user interface. In other approaches, even though the data is visually displayed, the patterns found by the user must be noted by the user, and then transformed, by the user, into modifications of the input parameters to the DM algorithm.

III. The Differences Between Discovery And Search

We use the term *discovery* to mean the process of finding causal relations between entities in the world. A causal pattern is one that describes the conditions, X, which, if they exist, mean that other conditions Y are true. This can be represented as an *if-then* rule:

$$if\ X\ then\ Y \tag{1}$$

The discovery of causal patterns is a well researched topic. For instance, in data mining (DM), this is referred to as finding *association rules*. Causal relationships can be automatically discovered by finding correlations or patterns among the fields of the data. Although correlation does not imply causation, in many techniques correlation is taken as an indication of causality. Data mining techniques such as rule induction or Bayesian networks try to automatically find causal relationships within the data.

Automatic discovery of causal rules is difficult due to the fact that most data is noisy and incomplete. With noisy data, spurious correlations may exist, and incomplete data does not provide enough information to find correlations. In many interesting cases (e.g. finding attack signatures in network logs), the environment is non-stationary, and the causal rules underlying the environment are changing as well; thus making automated learning difficult. Due to these difficulties, discovery of causal rules in domains with noisy, incomplete data or a non-stationary environment, is still best done by humans. This is the case for determining attack signatures in network intrusion detection.

Given a causal relationship, we can assume that if the antecedents (e.g. X as defined in equation 1), hold, then the consequent (e.g. Y as defined in equation 1) also holds. For instance, if we have discovered the rule *if many umbrellas are being sold, then it must be raining outside*; whenever many umbrellas are sold, we can infer that it is raining outside. To infer that conclusion, we must be able to determine that many umbrellas are being sold. We use the term *searching* to denote the process of determining whether X holds in some set of data. Searching is a task well suited to machines.

A. Discovery and Search in Intrusion Detection

A major task of security engineers is to detect and respond to intrusions within their networks. Intrusion detection has become a significant and active subfield of computer security research. Since there are vast amounts of data about the network, automatic intrusion detection techniques and systems have been developed. There are two types of Intrusion Detection Systems (IDSs): Signature and Anomaly based systems. Signature based systems such as Snort [6] contain a database of attack *signatures*. The systems then match network logs to the signatures and generate alarms. For instance a sample Snort rule is:

```
alert tcp any any ->
192.168.1.0/24 111
 (content: "|00 01 86 a5|"; msg: "mountd access";)
```

This rule generates the message "mountd access" for any packets processed with the destination address 192.168.1.0/24, destination port 111, and content "$\|000186a5\|$". This rule can be alternatively described as a causal rule of the form:

$$\begin{aligned} if \quad & DstIP \in 192.168.1.0/24\ and \\ & DstPort = 111\ and \\ & content = |000186a5| \\ then \quad & \text{“}Possible\ mountd\ access!\text{”} \end{aligned}$$

Signatures can all be reduced to such causal rules, defining what activity indicates an attack. Thus, signature based intrusion detection systems are just searching—they do not create signatures, but rather the signatures must be supplied. Usually, the signatures are created by human security engineers who have seen the attack and can discern the salient properties of it. Creating IDS signatures is the task of discovering causal relationships in the data.

Visualizations allow large amounts of data to be represented concisely. By transforming the data into a visual image, patterns are often more easily found in the data. Figure 1 contains coordinates of a circle drawn in a two dimensional space. But any human would be hard pressed to

ISBN 0-7803-9814-9/$10.00

x	y	x	y
1	20	18	8.7
7	18.74	6	19.07
20	0	−4	19.7
12	16	5	19.36
3	19.7	10	17.32
4	19.59	9	17.86
8	18.3	14	14.28

Fig. 1. Points lying on a circle. From this vantage point, it is hard to determine that this table represents a circle.

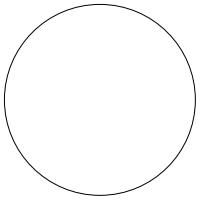

Fig. 2. Visualization of Figure 1

discover the circle pattern in Table 1. On the other hand, if the data was visualized, as in Figure 2, it is quite easy to see the circle.

Similar to the circle example, security visualizations allow security engineers to view large amounts of network data and find causal relationships. Tools like NVisionIP [7], VisFlowConnect [8], PortVis [9], and "The Spinning Cube of Potential Doom" [5] provide concise, rich descriptions of the data that enable security engineers to discover attacks and the signatures of the attacks.

Security visualizations are powerful tools that aid humans in discovering patterns, but they are currently limited by the fact that they provide no mechanism to search through the data—except of course allowing a human to manually sift through large volumes of binary or ascii data. The issue is that while visualizations are rich and highly detailed, they are not symbolic; that is, it is very difficult for a machine to manipulate these visual patterns and use them to search through data. On the other hand, if the visual pattern could be represented in symbolic terms, then a machine would be able to search for the pattern in data.

The key problem lies in transforming a visual pattern into a symbolic one that can be used by a computer to perform searches. The transformation will be domain specific, but general principles apply over a wide range of visualizations. In our case study, we will illustrate our approach for the transformation of visual patterns to symbolic patterns in the context of NVisionIP, a NetFlow visualization tool intended to provide situational awareness for security engineers. We call this *closing-the-loop*, as we link the human back to the process of filtering the data.

To allow security engineers to both discover new patterns and search for these patterns, the following prerequisites

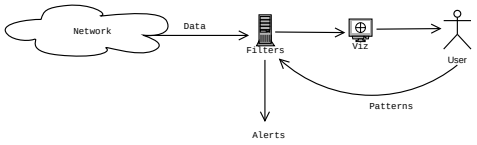

Fig. 3. Work flow in a system that implements closing the loop

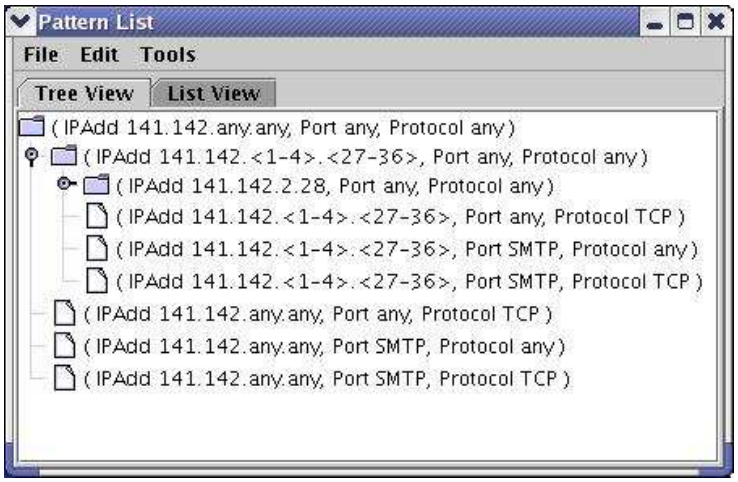

Fig. 5. The tree view of the pattern tree

are needed:

1. A rich visualization system for network data
2. User interaction with the visualization, leading to the discovery of visual patterns in the data
3. Transformation of the visual pattern into a symbolic pattern
4. The ability to search through a new set of network data using the symbolic pattern.

In this paper, we focus on (3) and (4). See [7], [8] to understand how we have addressed issues (1) and (2). Figure 3 illustrates the interaction between all four processes.

IV. A Case Study of Closing-the-Loop with NVisionIP

NVisionIP[7] is a visualization tool designed to provide situational awareness of a network for security engineers. Its aim is to represent transactional data taken from the routers on a network in a visual manner that is conducive for learning about both normal and abnormal behavior of a network.

Currently, NVisionIP uses NetFlows as its data source. A NetFlow record is an abstract representation of a sequence of packets transmitted between a source and destination host over a particular source and destination port pair. NetFlows keep track of the start and end time, source and destination port, number of bytes transmitted, number of packets transmitted, and IP protocol used (There are other optional fields, but these are common to all for-

ISBN 0-7803-9814-9/$10.00 ©2002 IEEE

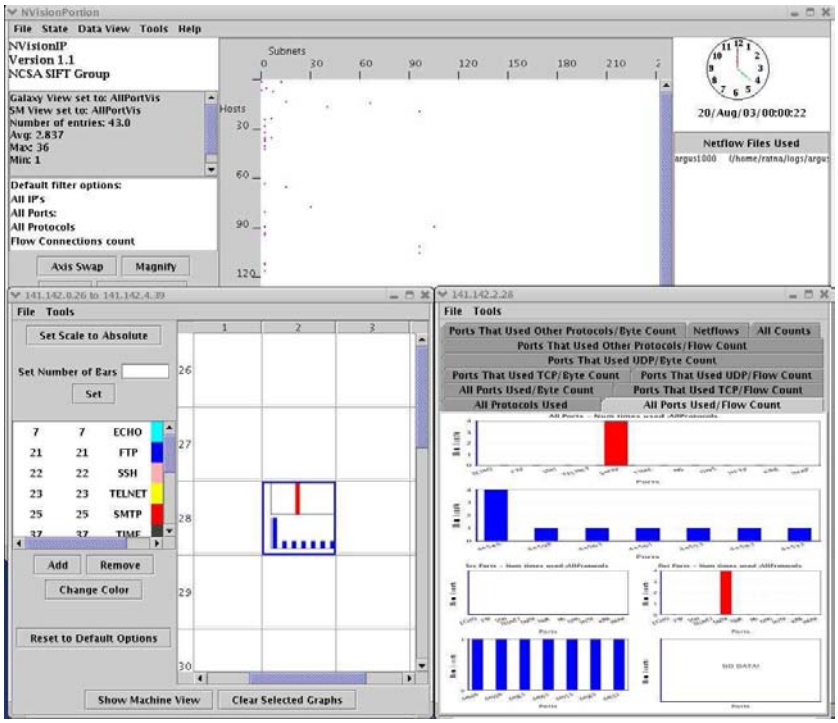

Fig. 4. NVisionIP screenshot. The background is the Galaxy View, the bottom right is the Small Multiple View, and the bottom left is the Machine View

mats). We will consider a flow to be comprised of the following seven fields:

Start Time	Start time of the flow
End Time	End time of the flow
Source IP	Source IP address
Destination IP	Destination IP Address
Source Port	Port on source IP address
Destination Port	Port on destination IP Address
Protocol	IP Protocol used for this flow.

NetFlows can be captured from various points on the network. For instance, CISCO routers can log NetFlow data, and there are also third party clients that can generate NetFlows from packets captured on an arbitrary host [10]

Figure 4 shows the three views of NVisionIP: Galaxy View, Small Multiple View and Machine View. The Galaxy View provides an overall view of the address space for an entire class B network. In the Galaxy View, the user can see statistics about machines, but cannot access more detailed information. The Small Multiple View shows more detailed information about smaller sets of machines, including information about port usage. Finally, the Machine View provides detailed information about a single machine, including the number of bytes transferred over a specific port, the types of protocols used and various other statistics.

For a more detailed explanation about NVisionIP, see our previous descriptive paper [7].

A. *What is a Pattern?*

In the context of NVisionIP, we consider patterns to be a set of constraints on the IP addresses, ports and protocols of a NetFlow record. A record is a match if its IP address, port number and protocol are within the specified ranges for each field. We will represent a pattern using this notation:

$$141.142. < ip_1 > . < ip_2 >, < portr >, < protr >$$

Where

ip_1	Range of subnets, like $< 30 - 40 >$.
	Can be $< any >$, signifying any subnet
ip_2	Range of hosts, like $< 35 - 90 >$.
	Can be $< any >$, signifying any host
$portr$	Range of ports, like $< 0 - 1023 >$.
	Can be $< any >$, signifying any port
$protr$	Range of protocols, like $< 6 >$.
	Can be $< any >$, signifying any protocols.

Searching for a pattern will involve finding all records in a set of NetFlows that satisfy the constraints of the pattern. A NetFlow record matches a pattern if:
1. The Source IP subnet and host or the Destination IP subnet and host are between ip_1 and ip_2 respectively. (Note: this kind of box like selection is exactly the same type of selection used for the SmallMultipleView of NVisionIP).

ISBN 0-7803-9814-9/$10.00 ©2002 IEEE

2. The Source port or Destination port are within *portr*
3. The protocol is within *protr*

For example, a pattern of:

$$141.142.<any>.<98>,<91-95>,<any>$$

will match with all records that have an IP address in the domain 141.142 with a host number of 98 that uses either port 91, 92, 93, 94 or 95.

The pattern:

$$141.142.<45>.<any>,<80>,<6>$$

will match with all records for hosts on the network 141.142.45.0/24 communicating on port 80 with protocol 6, i.e HTTP traffic.

B. Transforming User Behavior Into Patterns

We use the *Observer* paradigm to generate patterns from user behavior. In this paradigm, we imagine an observer peeking over the user's shoulder as she works with the tool—much like a word processor macro recorder. The observer will record the actions of the user, and store them as patterns. These patterns can be modified, output into a standard format or deleted by the user at any time.

The patterns created depends upon which view the user is currently focused on. In the GalaxyView, a pattern consists of a range of IP addresses. Whenever the user zooms in on a set of IP addresses, a pattern is created and placed in the pattern tree. The pattern created will be in the format of : $141.142.ipr_1.ipr_2,<any>,<any>$, for example, $141.142.<5-10>.<25-29>,<any>,<any>$.

In the SmallMultipleView, a pattern consists of a range of IP addresses. The user can select an entire row/column or a single machine. Based on the selection, the relevant pattern is generated and added to the pattern tree. Since the only actions in the SmallMultipleView involve choosing a subset of the machines, the patterns will only select a subset of machines as well. The pattern created will be in the format of: $141.142.<ipr_1>.<ipr_2>,<any>,<any>$. The main difference in using this view to create a pattern is that it is easier to select a very small or specific set of IP addresses because the view is already zoomed in on a smaller area.

In the MachineView the user can select specific ports and protocols by clicking on the bars in the histograms. This will create patterns that will specify the port and protocol portions of the pattern. For example, if the user is viewing the bar graph showing the number of times each protocol was used by machine 141.142.9.8, and the user clicks on the bar for TCP, the pattern $141.142.<9>.<8>,<any>,<TCP>$ will be generated. On the other hand, if the user is looking at the histogram showing the number of times each port was used by machine 141.142.9.8; and clicks on the bar for port 80, the pattern $141.142.<9>.<8>,<80>,<any>$ will be generated. The most specific pattern is created at the MachineView level, as the MachineView shows the most detailed information.

C. Pattern Tree

Many patterns will be generated via the observer method—many of which may not be useful—since the observer will be recording all activity. We organize the patterns by creating a *Pattern Tree* that shows how the patterns are related to each other.

The pattern tree will organize patterns from most general to most specific. The parent-child relationships in the pattern tree indicate varying levels of specificity and also indicate from whence the patterns were generated. Figure 6 shows a sample pattern tree. The root of the tree is the most general pattern possible, $141.142.<any>.<any>,<any>,<any>$. This corresponds to the GalaxyView before any selections. If the user picks a range (in this case a square from the 2-D grid) of IP addresses to view in the Small Multiple View, a new pattern $141.142.<ipr_1>.<ipr_2>,<any>,<any>$ is created and added as a child to the root node. Thus, the second pattern is more specific than the root node.

Within the Small Multiple View, if the user selects a single machine to bring up in the Machine View, the pattern $141.142.<subnet>.<host>,<any>,<any>$ is created and added as a child of the Small Multiple View pattern ($141.142.<ipr_1>.<ipr_2>,<any>,<any>$). Finally, if ports or protocols are chosen in the Machine View, these patterns are added as children to the pattern: $141.142.<subnet>.<host>,<any>,<any>$.

We can see that the parent pattern of a child shows from which view the child pattern was generated. When we add patterns that were generated in the Machine View, though, we add the port and protocol selections to all the nodes in the path from the Small Multiple View pattern to the root node—essentially creating aunt or uncle nodes. Figure 7 illustrates what happens when a pattern is added.

D. Representation of patterns in PatternList

Patterns created can be viewed in two different formats: Tree view and List View. The tree view always has a root node that holds the most generalized pattern that can be created using NVisionIP, i.e.: $141.142.<any>.<any>,<any>,<any>$.

D.1 Searching with the Patterns

The patterns generated by observing the user behavior can be transformed into the widely used *libpcap* format. After this transformation, these patterns can be used with well known tools like Snort, TCPdump or Ethereal to allow the security engineer to search through network traffic data. In this way, the symbolic representation of the patterns can not only be used to search through NetFlow logs

ISBN 0-7803-9814-9/$10.00 ©2002 IEEE

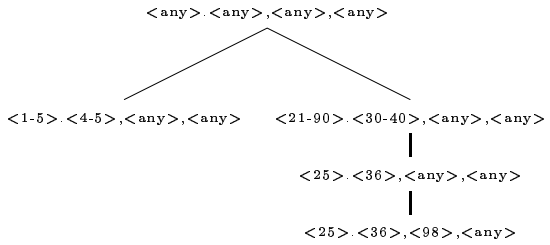

Fig. 6. Sample Pattern Tree. In this case, the user initially chose to view, in the Small Multiple View, the range of machines 141.142.<21-90 >.<30-40>. The user then chose machine 141.142.25.36 to view in the Machine View. While viewing the bar graphs, the user clicked on the bars for port 98 and then clicked on the bar for protocol 11. The left hand branch of the tree indicates that the user choose to view the range 141.142.<1-5 >.<4-5>in the Small Multiple View, but did not go any further from there.

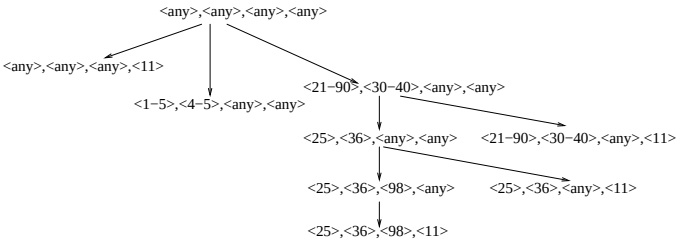

Fig. 7. Pattern Tree from Figure 6, after choosing protocol 11. Four new patterns are added to the tree, one for each node on the path from the parent node to the root node. In this method, all possible patterns and combinations of patterns are stored.

files but also many other types of network logs. Therefore, there are likely many unforeseen applications for our patterns due to the wide-spread use of libpcap filters.

V. Conclusions and Future Work

In this paper we have argued for the need to separate the tasks of pattern discovery and pattern matching in what we call closing-the-loop. This decoupling in security visualization tools will allow human operators to focus on the tasks at which they excel (pattern discovery) while allowing machines to focus on the operations at which they excel (pattern searching). We further describe our first implementation of this idea in our NetFlow visualization tool, NVisionIP.

We believe that recent enhancements we have made to NVisionIP is the first step in a fundamental change to security visualization tools. A logical next step would be to more fully incorporate this searching into NVisionIP itself. In this way the searches would dynamically change the data being viewed in NVisionIP, or they could be used to process raw NetFlow logs. Further into the future we would like to visually create firewall rules or rule-sets that change how and what is collected at the network sensors.

The possible applications of this concept of closing-the-loop are almost limitless. For example, while it would be

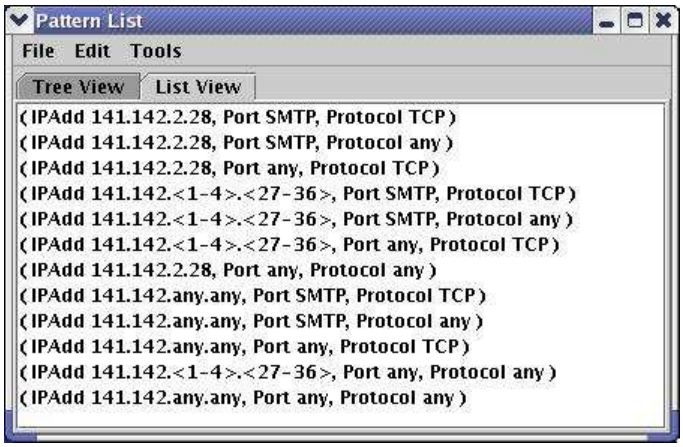

Fig. 8. The list view of the pattern tree

difficult to create IDS rules for Bro or Snort with NetFlow data—NetFlows contain transactional data only and no packet information—tools visualizing other datasets could conceivably implement this idea by visually generating such rule sets. Another idea is to create a tool that allows users to select parts of the IP address space on which to run rudimentary data mining algorithms that look for correlations based on patterns specified for by the user. The results can then be visually presented back to the user who can choose which machines to investigate in even further detail with more complex algorithms.

References

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[2] M. Dodge and R. Kitchin, *Atlas of Cyberspace*. Harlow: Addison Wesley, 2001.

[3] G. Conti and K. Abdullah, "Passive visual fingerprinting of network attack tools," in *Proceeding of the CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC) 2004*.

[4] S. T. Teoh, K.-L. Ma, S. F. Wu, and X. Zhao, "Case study: Interactive visualization for internet security," in *IEEE Visualization*, 2002.

[5] S. Lau, "The spinning cube of potential doom," *Communications of the ACM*, vol. 47(6), pp. 25–26, 2004.

[6] www.snort.org.

[7] K. Lakkaraju, W. Yurcik, A. J. Lee, R. Bearavolu, Y. Li, and X. Yin, "Nvisionip: Netflow visualizations of system state for security situational awareness," in *Proceeding of the CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC) 2004*.

[8] X. Yin, W. Yurcik, M. Treaster, Y. Li, and K. Lakkaraju, "Visflowconnect: Netflow visualizations of link relationships for security situational awareness," in *Proceeding of the CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC) 2004*.

[9] J. McPherson, K.-L. Ma, P. Krystok, T. Bartoletti, and M. Christensen, "Portvis: A tool for port-based detection of security events," in *Proceeding of the CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC) 2004*.

[10] "Argus web site." http://www.qosient.com/argus/index.htm.

ISBN 0-7803-9814-9/$10.00