Visualizing Firewall Configurations Using Created Voids

Shaun P. Morrissey Georges Grinstein

Institute for Visualization and Perception Research
University of Massachusetts/Lowell

ABSTRACT

Security configuration files are created and edited as text files. These files are the essential definition and control of the behavior of security devices. Despite their significant size, complexity, and the possibility of interaction between entries, no visually sophisticated tools exist that explicitly capture and visualize problematic interactions between rules to aid in the comprehension and modification of configuration files. Our initial work on the direct visualization of firewall configurations showed the limitations of visualizing just the range of packets that can be accepted. To visually capture the interactions between rules, we introduce the concept of a "created void." Created voids capture the information about destructive interactions between rules in a firewall ruleset, where an overlap between a deny rule prevents that packet from reaching an accept rule later in the ruleset. We present a lossless five-dimensional visualization of the convex solid decomposition of the set of acceptable packets from a firewall configuration, augmented with visual representations of created voids. This interactive visualization is embedded in a simple firewall ruleset editor, allowing the user to investigate the effect of changes in the ruleset.

KEYWORDS: Firewall visualization, network security, firewalls, filtering routers, security configuration, created void.

INDEX TERMS: C.2.0 [**Computer-Communication Networks**]: General – *Security and protection (e.g., firewalls).* H.5.2 [**User Interfaces**]: *Graphical user interfaces (GUI).*

1 INTRODUCTION

At present all security configuration files are created and edited as text files, despite their significant size, complexity, and the possibility of interaction between entries. Text is acknowledged to be the least efficient (lowest bandwidth) use of the visual channel into the brain. Clearly visual representations of security configurations could improve comprehension, and thus correctness, of the security configuration. We describe the development of this concept and illustrate it with a relevant use case for firewall administration, the failure of a firewall to permit a desired network interaction. This particular case is drawn from a real example of a problem at a company with satellite offices embedded in the offices of its clients, but it is representative of a wide range of issues faced by firewall administrators. In addition, two reports that focus on interactions between rules and provide example rulesets have been selected as a basis for initial analysis of the technique [1][7].

We briefly describe the approach used to convert order-dependent firewall rulesets into a static acceptance set using the concepts of constructive solid geometry in a five-dimensional space. We describe the theory and technique of the modification to the mechanics of penteract interaction to enable the use of created voids as visual elements. We also describe the use of a parallel coordinate representation [3] for our initial and preliminary display approach for the convex solid decomposition of the acceptance volume. We note where this display does and does not capture the classes of rule interactions identified by Al-Shaer and Hamed [1]. We also note where the parallel coordinates representation does not appear to clearly present certain relationships within the data, and identify our initial candidate for a representation that may capture such relationships at the cost of stepping away from a lossless representation.

It is important to note that this effort is an initial approach to visualizing the configuration of a firewall, and that it is not a visualization of the traffic that has passed through a firewall. Most references to firewall visualization, e.g. Marty's *Applied Security Visualization* [4], are visualizations of traffic that has been accepted, with some connection to the rule that decided the fate of the packet.

The PolicyVis tool [6] is an attempt to display configuration. PolicyVis presents selected projections of subspaces of the configuration based on a specific user query. PolicyVis relies on operator selection of the initial visual dimension, and then detection of a visual overlap to detect interactions. It also uses a restricted set of text cues to convey the protocol dimension, limiting it to two values (TCP and UDP). It does not explicitly calculate and retain the interactions that cause changes in the acceptance volume, the created voids approach discussed herein.

2 SCOPE: ABSTRACTED FIREWALLS AND EDITING

The software artifact that serves as a basis for the work is a simplified editor for firewall rulesets. This has been augmented by visualizations of the acceptance space and interactions between the visualizations and the editing frames. The software is a Java application using the Swing GUI class library.

Our initial focus on the visualization of security configuration files is on firewalls or access control lists that filter Internet packets based on values in five packet header fields. The dimensions we consider are source address (SA), source port (SP), protocol number (PR), destination port (DP) and destination address (DA). This simplified abstraction serves as the basis for much of the research literature in the firewall arena.

The space of all possible packets is an integer lattice in these five dimensions. This space is referred to herein as the *acceptance space*. The *acceptance volume* for the particular configuration is the subset of the acceptance space that corresponds to all the packets that the firewall would allow to pass. Note that this is not a network traffic visualization, and in particular not the set of packets that have passed through the firewall in any period. The object of visualization is the acceptance volume as embedded in this acceptance space.

Department of Computer Science,
One University Avenue, Lowell, MA 01854
smorriss@cs.uml.edu, grinstein@cs.uml.edu

6th International Workshop on Visualization for Cyber Security 2009
11 October, Atlantic City, New Jersey, USA
978-1-4244-5415-0/09/$25.00 ©2009 IEEE

As a first approach to visualization of the five dimensional objects that make up the acceptance volume, we selected a parallel coordinate representation. This choice allows for a lossless representation of simple convex connected volumes in the five-dimensional space.

3 SOFTWARE DESCRIPTION

The major components of the software are a table-based editor for the rules, the calculation of the acceptance volume, and the visual representation of the acceptance volume. In addition, various interactions are supported to enable interactive investigation of ruleset changes. The editor is simply described, but the mechanics of the visualization will be handled in subsections.

The editor uses a table that accepts ranges in each of the five variables. Separate lists of hosts, port ranges, and protocol ranges are maintained to allow the user to assemble rules by picking from lists. The user can turn rules on and off to see the effect on the acceptance space and can edit each of the fields of the rules with immediate feedback through the visual display.

Most commercial firewall products allow an "individual" rule to incorporate multiple source ranges, multiple destination ranges, and in some cases, multiple services. At this stage, our software only permits an individual rule in the ruleset table to have a single source address range, a single destination address range, and single ranges in the source port, destination port, and protocol variables. This would require one commercial rule to be mapped to *m* x *n* x *d* rules in our editor, where *m* and *d* are the number of multiple hosts appearing as sources and destinations, and *n* is the product of the number of ports and protocols used in the rule.

3.1 Acceptance Volume Calculation

Firewall or ACL (access control list) rulesets use order-dependent semantics. Each rule specifies the set of packets it applies to in a predicate, and an action to accept or deny. The packets are compared to the rules in sequential order. The first rule whose predicate matches the packet takes its designated action, and then the process continues on to the next packet. This allows for the possibility of interaction between the rules, intended or unintended, due to partial or complete overlaps between their predicates.

The possibility of interaction is not automatically a problem. The order-dependence is sometimes used to compactly represent an enterprise policy. For example, if most of the Internet is to be denied access to a particular server, with the exception of one or two particular other entities, then this can be encoded in three rules with the last rule forbidding access to all, a deny rule. Preceding it in the ruleset are the two accept rules, each allowing a specific entity in to the server.

For the purposes of visualization, the order-dependent representation in the ruleset must be converted to a static acceptance volume. Guttman [2] published a recursive procedure for converting the ruleset to the static acceptance volume using simple set operations. The procedure treats the predicate of each rule as a volume in the acceptance space and adds these to the acceptance volume of the firewall using the set operations of union and set-difference. The result is a set of points in the acceptance space that correspond to all packets that could pass through the firewall.

We have implemented an equivalent [Guttman, personal communication 2009] iterative version of this algorithm as a restricted case of constructive solid geometry (CSG) in five dimensions. [5]

We modified the normal calculations of the CSG to retain information on sections of the acceptance space that are accepted by parts of the ruleset but rejected by others that take priority.

3.1.1 Algorithm

As mentioned previously, the space of all possible packets is an integer lattice in five dimensions. Each elementary rule (predicate) in the ruleset is treated as an interval in each of the five dimensions of the acceptance space (lower and upper limits). This is a special case of a penteract, a five-dimensional orthogonal hyper-solid with all of its edges and faces aligned with the coordinate axes of the acceptance space. All of the volumes of interest can be created by the union and/or set-difference of these penteracts [2].

For two reasons, visualization and software tractability, we implemented the convex solid decomposition of these CSG operations, rather than the general operations for arbitrary shapes or volumes. The use of the convex solid decomposition (CSD) simplifies the data structure required to support CSG in five dimensions. This also allows the definition of procedures for union and set-difference to be implemented one dimension at a time and then repeatedly applied to each dimension. This leads to a simpler code structure with greater assurance of both algorithmic and implementation correctness (a reduction to 13 cases applied five times as opposed to over 371,000 cases for the possible intersections between two penteracts). It does have the side effect of multiplying the number of penteracts that must be handled by the visualization. The intersection between two rules or penteracts can add up to 11 penteracts to the convex solid decomposition of the acceptance volume, depending on the nature of the overlap.

The iterative algorithm implicitly assumes that the final rule in a ruleset is a "deny all" rule in keeping with the generally accepted firewall policy that what is not specifically accepted is denied. Thus, the starting acceptance volume is the null set.

The software starts from the last rule and proceeds to the first. At each rule, if the rule is an accept rule, the penteract describing the predicate of the rule is incorporated into the acceptance volume by set union. If the rule is a deny rule, the penteract representing the deny rule volume is subtracted from the acceptance volume, a set-difference operation.

We comment on certain specialized aspects of the set-difference operation in the next section.

The implementation maintains a convex solid decomposition at all stages. This allows all volumes to be built up by lists of simple axis-aligned penteracts rather than more complex geometric structures.

3.1.2 Penteract Constructive Solid Geometry and Created Voids

There are two specific aspects of the mechanics of penteract interaction that require comment. The first issue deals with those decisions that have been made to increase confidence in the correctness of the software. The second are the specific differences required to handle created voids.

The implementation of the set-difference operation takes advantage of an identity. For the acceptance set A, and the subtracted penteract B, where A-B is the set difference, we have:

$$A - B = A - (A \cap B)$$

Thus, the subtraction of B from A is only the subtraction of the parts of B that overlap A. This reduces the problem of calculating the set difference to determining the overlap of the subtracted penteract with the set of existing penteracts in the acceptance volume. All operations can be reduced to a collection of the interactions of two penteracts.

The technical point that simplifies the code and enables the development of created voids is that the CSD of the interaction of two intersecting penteracts always produces three lists of penteracts. For a targeted penteract in the acceptance volume, T,

and an added penteract A (from the next rule being processed), the result will be a list of penteracts that were only in T (the set T-A), another list of penteracts that were only in A (the set A-T), and a single penteract that is the intersection of A and T (the set T ∩ A). All set operations can be regarded as simply different dispositions of these three parts, as shown in Table 1.

Table 1. Disposition of Penteract Lists for Set Operations

Operation	T-A	T∩A	A-T
union	keep	keep	keep
intersection	discard	keep	discard
difference	keep	discard	discard
void difference	keep	retain but mark as void	discard

This allows the development of a single interaction method that returns the three lists and wrapper methods that convert the lists to results for specific set operations.

For most commercial firewalls, there is some compilation of the ruleset into a data structure that enables fast determination of whether a packet will be accepted. The result of that focus on acceptance is that information that relates to packets accepted by a late rule in the ruleset but denied by an earlier rule is simply discarded. An active firewall doesn't need to know about possibilities, only the final decision. This corresponds to using the standard set difference disposition shown in Table 1.

In our software, the void-difference is used instead. To illustrate the point we use a three-dimensional model. The penteract that is the overlap, shown in blue in Figure 1, is discarded if Rule A (red with top made transparent to show intersection volume) is a deny rule being subtracted from Rule B (green).

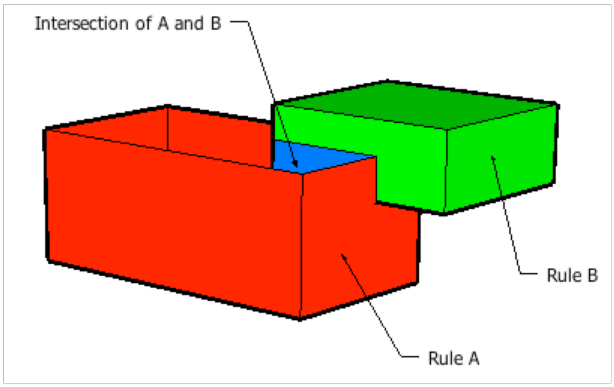

Figure 1. Three-dimensional Example. The penteract that is the overlap, shown in blue is discarded if Rule A, shown in red with its top made transparent to reveal the intersection volume, is a deny rule being subtracted from rule B shown in Green.

In our software, the penteract shown in blue is retained as an object in the acceptance volume list. All penteracts carry a Boolean flag, true if the volume is a solid, a part of the acceptance space, and false if the volume is a void carved out of an accepted volume. This retains the information about the interaction with the overlapping rule. It allows us to present that void as a visual object allowing the user to see the source of a failure to pass packets.

It is now possible to detect that multiple rules affect the same volume such as when another interaction with an accept rule occupies the same space.

3.1.3 Rule Provenance of Penteracts

We augment the information carried by each penteract in one other way to support interaction with the editor. Each penteract maintains a list of rules. When penteracts are created during the calculation of the acceptance volume they come from a specific rule. That rule starts the list.

Whenever two penteracts overlap, the overlapping penteract is separated in the convex solid decomposition. The lists for the overlapping (intersection) penteract are the concatenation of its parents. In Figure 1, the red penteract carries rule A in its list. The green penteract carries rule B. The blue penteract carries both rule A and rule B in its list. This behavior links every active volume in the acceptance space, void or solid, to the set of rules that created it. In turn, this enables a selection operation on the graphical side to highlight the rules in the editor that led to the creation of that component of the acceptance volume.

3.2 Visualization

The visualization uses a parallel coordinate representation [3], with the upper and lower limits of each dimension defining the boundaries of a ten-sided polygon in what can be called a parallel set enclosure. Parallel coordinates are a lossless representation selected under the general principle that allowing the occlusion of data in an initial visualization approach is risky. Generally only subject matter experts can tell you that occlusion is tolerable and these visualizations are not yet developed to the point of supporting user testing.

The example below shows a rule allowing a particular Class A address block to have http: access through the firewall to a block of servers. Here, http access is defined as the 3-tuple of any source port, protocol number of 6 (TCP), and destination port 80.

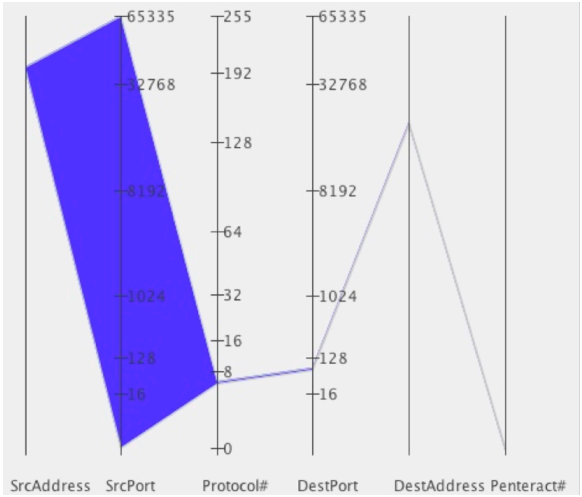

Figure 2. Rule allowing a Class A address access to an HTTP server

This visualization approach cannot represent penteracts with pieces carved out of them (non-convex) but does provide a complete representation of a convex solid axis-aligned penteract. Hence, the visualization also drives use of a convex solid decomposition of the acceptance volume.

The axes for protocol number and source/destination port use a non-linear mapping to visually highlight the lower values, as most rules specify values in the lower part of the range of these values. The source address and destination address use a linear scale, converting IP address fields to an unsigned 32-bit integer. Given only about 400 pixels in the vertical part of the display, this essentially limits the resolution to the 256 Class A addresses. The Penteract# axis is a visual display of the index of the penteract in the list of penteracts. Its sole purpose is to give a unique bit of screen real estate to each penteract.

Acceptance penteracts are rendered in blue, voids in red. A transparency factor (typically around %80) is used to give some indication of overlap. The edges are drawn over the filled shapes in slightly different colors to help show overlaps.

Selection, by clicking, causes a change in color scheme in both fill and edge. Selecting one or more penteracts in the display highlights the affected rules by changing the display of the rule number column in the table editor for the rules.

There are controls for imposing a window on the number of penteracts displayed (a spinner) and which part of the list is displayed (a slider). Checkboxes control the display of accept volumes, voids, or both.

4 Case Study

One of our more interesting use cases is where a legacy rule combined with an external over-configuration to prevent email service between two offices. At one point the enterprise had a server in a demilitarized zone offering a service for internal use only on port 32760. To prevent outside use, rules were put in place that prevented use of port 32760 across the perimeter as either source or destination port. A decade later the rule was in place but the server was not. The distant office suffered from a too-strictly configured email server, where the DNS query that precedes the actual email transfer was hard-wired to use 32760 as a return port. The result was that the DNS query never completed and no email transfer could succeed. Finding the rule that caused this problem took days.

Figure 3, below, shows the display resulting from a ruleset contrived to show this same phenomena. Here six total penteracts are generated with two voids because the 32760 deny rule cleaves both the DNS accept and the SMTP accept rule.

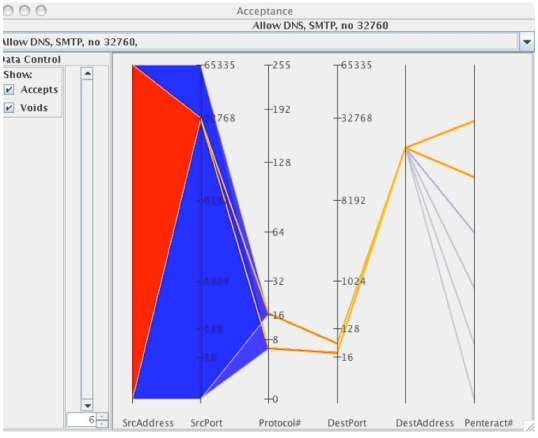

Figure 3. DNS and SMTP Created Voids. Six total penteracts with two voids created by the interaction of the 32760 deny rule's cleaving. The two voids are contained within the remainder of the DNS and SMTP accept rules (details in text)

The example clearly shows the one of the two voids created by the interaction. The other is obscured due to overlapping values in the two void ranges. It is possible to work out that the two voids are contained within the remainder of the DNS and SMTP accept rules, but it is not immediately obvious. This example also shows that the transition from single n-dimensional points to enclosures in five dimensions resurrects the issue of visual loss of data.

Our next example illustrates two additional types of results. Al-Shaer and Hamed [1] developed a complete description of the possible overlaps between firewall rules. They included a dataset in their article that compactly included and demonstrated all of the

Table 2. Ruleset of Al-Shaer and Hamed

	Protocol	Source		Destination		Action
		Address	Port	Address	Port	
1	tcp	140.192.37.20	any	*.*.*.*	80	deny
2	tcp	140.192.37.*	any	*.*.*.*	80	accept
3	tcp	*.*.*.*	any	161.120.33.40	80	accept
4	tcp	140.192.37.*	any	161.120.33.40	80	deny
5	tcp	140.192.37.30	any	*.*.*.*	21	deny
6	tcp	140.192.37.*	any	*.*.*.*	21	accept
7	tcp	140.192.37.*	any	161.120.33.40	21	accept
8	tcp	*.*.*.*	any	*.*.*.*	any	deny
9	udp	140.192.37.*	any	161.120.33.40	53	accept
10	udp	*.*.*.*	any	161.120.33.40	53	accept
11	udp	*.*.*.*	any	*.*.*.*	any	deny

possible overlap relationships. It is included here as Table 2, with slightly different formatting.

The resulting display from this dataset is shown in Figure 4. This figure shows 103 penteracts, demonstrating a significant need to provide zooms or multiple views or selection into the data. The scrollbar and spinner shown to the left of the penteract display allow the user to limit how many penteracts are displayed, and to move that window over the list of penteracts. This has highlighted the need to be able to selectively view penteracts "related" to one another. The current multiple views of the data use the list of penteracts as a basis for display. However, the order of insertion into the list does not guarantee proximity between related penteracts.

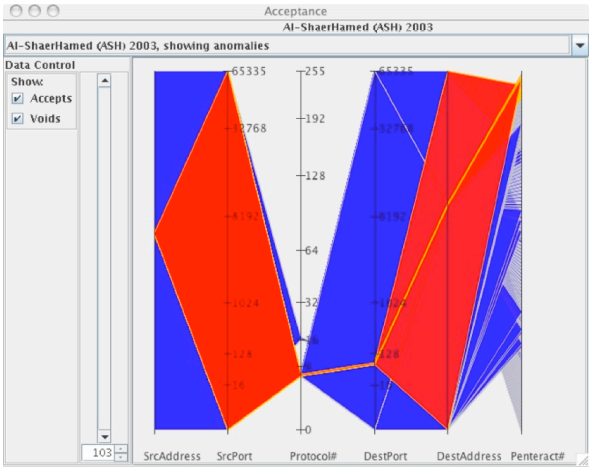

Figure 4. Visualization of Al-Shaer and Hamed Ruleset. 103 penteracts are presented clearly highlighting the need for interactive data zooming and multiple views.

What is necessary is a multiple view control for the display that selects a subset of the rules and limits the display to those

penteracts whose provenance list includes the selected rules. This will specifically support focusing on interactions between selected rules.

5 Conclusions, Assessment, and Directions

Occlusion in the parallel set enclosure view becomes an issue even at relatively small numbers of rules. The increase in the number of penteracts as the size of ruleset increases demands the addition of various support visualization and analysis tools including, for example, data zooming and multiple views. These tools will support selecting data subsets that meet a set of relatedness or relevance criteria.

The current display represents a preliminary proof-of-concept for exploring the utility of created voids. Modification of the display to have special cases, such as a value of "any" in a predicate field, mapped to a point instead of the full range would decrease the visual overlaps on the display and have the benefit of compactly representing dimensions where two penteracts completely coincide. It is less clear that the user would immediately comprehend that every other penteract with a restricted range of values on an axis was also overlapping with all of the penteracts drawn through the "any" point.

The parallel coordinate representation does not appear to clearly show the "containment" of one volume or void within another.

We are developing an alternative interactive 3D representation

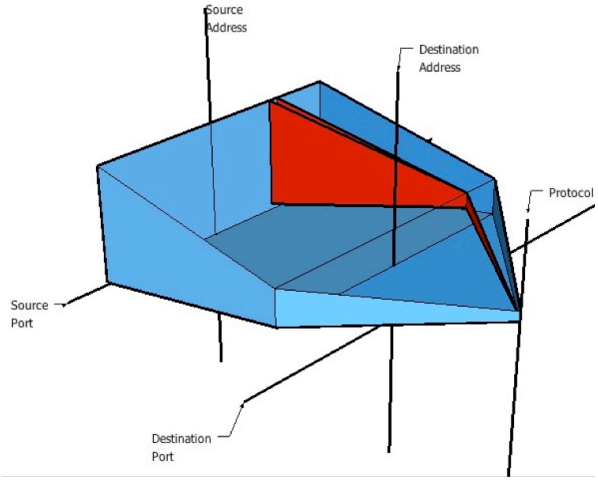

Figure 5. 3D Representation. The plane in the left of the object represents the source address and source port axes. The destination address and destination port are similarly set up as a two-dimensional plane. The remaining value, the protocol number, is treated as a single axis. The top surfaces of the accept penteract (blue) has been rendered transparent to display the interior. A deny rule partially overlaps the accept rule penteract. Its smaller range in both the source plane and the destination plane carve out an interior volume with respect to the accept penteract that corresponds to packets that will be denied by the firewall configuration (rendered in red).

using OpenGL, and the design is shown in Figure 5. In this representation, the source address and source port axes are treated as a two-dimensional plane (located to the left in the figure). The destination address and destination port are similarly set up as a two-dimensional plane. The remaining value, the protocol number, is treated as a single axis.

A range of source addresses combined with a range of source ports defines a rectangle in the source plane. In a similar way, the range of destination addresses and destination ports defines a rectangle in the destination plane. The range of protocol values allowed by the penteract determines an interval along the protocol axis.

The polyhedron defined by the rectangle in the source plane, the rectangle in the destination plane, and the triangular section converging to the upper and lower limit on the protocol axis, contains all of the packets that are accepted by that penteract. In the figure the top surfaces of the accept penteract (shown in blue) has been rendered transparent to display the interior. In this design example case, we have a deny rule that has partially overlapped the accept rule penteract. Its smaller range in both the source plane and the destination plane carve out an interior volume with respect to the accept penteract that corresponds to packets that will be denied by the firewall configuration. Here the denied volume is rendered in red.

This representation visually corresponds to a pipe or opening in the firewall, allowing the set of packets corresponding to its interior to pass through the firewall. The visual metaphor has a certain appeal, matching a simple physical construct.

Our next steps are to modify the above interactive visualizations of rulesets to include clearer representations of the enclosures and overlaps of rules to further improve comprehension of firewall configuration.

References

[1] E.S. Al-Shaer and H.H. Hamed, Firewall Policy Advisor for anomaly discovery and rule editing, *IFIP/IEEE Eighth International Symposium on Integrated Network Management*, pp. 17 – 30, 24-28 March 2003. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1194157&isnumber=26863

[2] J.D. Guttman, and A.L. Herzog, Rigorous automated network security management. *International Journal of Information Security*, 4(1-2), 2005.

[3] A. Inselberg and B. Dimsdale, Parallel coordinates: a tool for visualizing multi-dimensional geometry. *Proceedings of the First IEEE Conference on Visualization*, 1990. vol., no., pp.361-378, 23-26 Oct 1990. URL=http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=146402&isnumber=3914

[4] R. Marty, Applied Security Visualization. 1st ed., Addison-Wesley Professional, 2008.

[5] M.E. Mortenson, Geometric Modeling, 3rd ed., Industrial Press, 2006.

[6] T. Tran , E. Al-Shaer, and R. Boutaba, PolicyVis: firewall security policy visualization and inspection, *Proceedings of the 21st conference on Large Installation System Administration Conference*, p.1-16, November 11-16, 2007, Dallas, TX, USA

[7] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra. FIREMAN: A toolkit for FIREwall modeling and ANalysis. *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, IEEE 2006.