A Visual Mathematical Model for Intrusion Detection

Greg Vert Deborah A. Frincke Jesse C. McConnell

Center for Secure and Dependable Software
Department of Computer Science
University of Idaho
Moscow, Id. 83844-1010

Abstract

To balance the examination of large quantities of data with the difficulty of comprehending such quantities, we propose a geometric approach to detection enhanced by a visual component. Visually presented information can encode large amounts of complex, interrelated data, can be quantified and manipulated, and is something that human beings naturally process well. Limitations of traditional intrusion detection system (IDS) techniques are as much a function of the ability of a human to process large amounts of information as they are limitations of the techniques themselves. In this paper, we present a discussion of a geometric model which appears to be useful in performing many of the activities used in traditional intrusion detection while including a useful visualization.

1 Introduction

Intrusion data has traditionally been presented in text form and, given the typical volume, is difficult for security officers to process. Limitations of traditional intrusion detection techniques are as much a function of the ability of a person to process large amounts of information as they are limitations of the techniques themselves. "To reduce computational requirements to a realistic level, intrusion detection systems focus on a limited set of system and user attributes." [HDL+90] Dynamic behavior of a hostile user across a networked system is hard to encapsulate and represent with such a limited set. In order to balance the need to examine large quantities of data with the difficulty of comprehending such quantities, we propose a visual geometric approach to detection. Geometric models lend themselves to combination and prediction. Visually presented information can encode large amounts of complex, interrelated data, can be quantified and manipulated, and is something that human beings naturally process well. Two things are needed in order to improve the performance of an IDS.

- A standard by which to encode and represent complex computer systems.

- A visual model presenting information in a context which data may be presented as relationships between components.

In this paper we present a discussion of a useful geometric model for performing several of the activities used by traditional intrusion detection systems while including a useful visualization.

2 The Visualization Model

A spicule is a spheroid geometric primitive which we use as the basis for our visualization model. One spicule represents the system model for one computer on a network. The spicules volume is determined by the a measure of the security fitness of a computer. Security fitness is a weighted sum of factors that add or detract from the vulnerability of a computer. Computer systems with a high security fitness will have smaller spicules than those of higher risk. Since the security fitness can be represented in this spherical fashion, it enables us to use the volumes and radii of a spicule as a mathematical property.

Spicules also model other features of system activity, viewing them in terms of vectors. The values of some of these features, called tracking vectors, can be normalized between zero and one hundred percent. An example of a tracking vector is CPU utilization, which can not exceed one hundred percent. As a featured value grows, its associated tracking vector travels along a path on the surface of the spicule towards the vertical axis. Vectors which can not be normalized in this fashion are called fixed vectors. These vectors are located at the equator of the spicule and can grow in magnitude coplanar to the horizontal axis. An example of a fixed vector is the number of child processes forked by a particular user. A signature can be obtained from both tracking and fixed vectors by tracing the path from a starting state to an end state. This signature can in turn be used as a mathematical property.

Security fitness, fixed vectors and tracking vectors visualize current approaches to intrusion detection in the following ways.

- Considering a normal system state, the angles and magnitudes of any vector can be established as a base case. The base case can further be used in anomaly detection.

- A geometric signature can be obtained by using a set of tracking vectors with characteristic angles or fixed vectors with magnitudes for either normal or attacked systems.

- Spicules can use ranges for vectors to account for normal operation. Under certain intrusions, these vectors will pull away from their normal ranges indicating unusual behavior.

- “A spicule can model effectively and easily up to 360 variables in a single state at a given point in time. Because it is geometric, it is simple to add or subtract spicules.

This change form of the spicule is a tessellation." [Gas83]. State machines have often been used in intrusion detection for state transition analysis. A similar concept is used in anomaly detection for predictive pattern matching. The spicule can be viewed as a state machine and duplicate these techniques with geometric and vector manipulation.

2.1 A Closer View of the Spicule

Spicules change over time and the notation $A(t_i)$ represents the spicule for computer A at time t_i. Figure 1 shows a spicule from both a polar and side view.

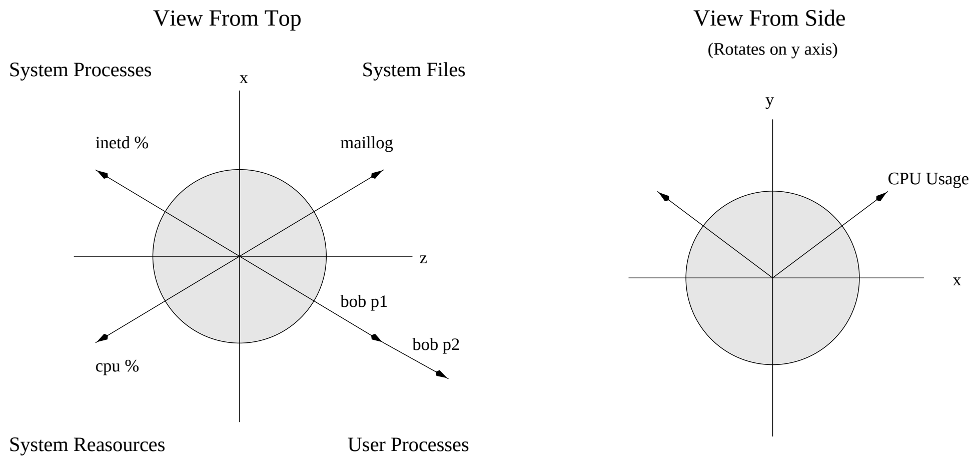

Figure 1: Polar and Side View of a Spicule at t_0

Four information classifications, represented by four quadrants around the spicule, are included in Figure 1: system resources, user processes, system files and system processes. The vectors in these quadrants may vary in number and type depending upon the information being represented. For intrusion detection, an appropriate set of feature vectors is dependent upon the system characteristics useful in detecting attacks.

The surface of a spicule is covered by feature vectors representing different system aspects. Every unit vector is of the same length, but in the case of user processes the vectors can be added, i.e. Bobs' p1 and p2 of Figure 1. Initially all vectors are positioned in a resting position on the horizontal axis. Various auditing tools run continuously on a computer collecting data which is used dynamically to render and update the spicule. The following is the process for updating a spicule.

1. The tracking vectors moving towards the vertical are shown as the percentage of their maximum value. For example, if CPU usage was 50% at time t_1, the vector representing CPU usage would be seen at 45°, along a track to the vertical axis of the spicule.

2. Vectors can combine within a quadrant based on the highest rates of change over a single time unit. For example, assume the highest rate of change in the System Resources

quadrant is CPU usage, CPU throughput, and Sendmail traffic. We can combine those three vectors with the highest rate of change using vector width as an indicator of the amount of change. The resultant vector would look similar to the one in Figure 2.

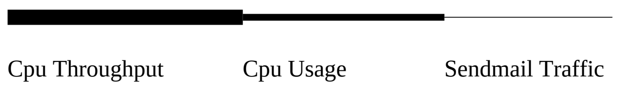

Figure 2: Recombined Feature Vector

Vector recombination provides additional signatures of system operation. In conjunction with the precession of vectors along the vertical axis and 360 potential features to be monitored, a unique model can be visualized and measured for normal and abnormal system operation. Figure 3 shows a spicule utilizing this at time t_n.

In this sample spicule we see:

- vector recombination by quadrants and vector signature development
- vertical axis which represents the maximum activity of each feature vector
- development of a set of characteristic angles $\{o', o'', ...\}$ for the spicule of in Figure 3

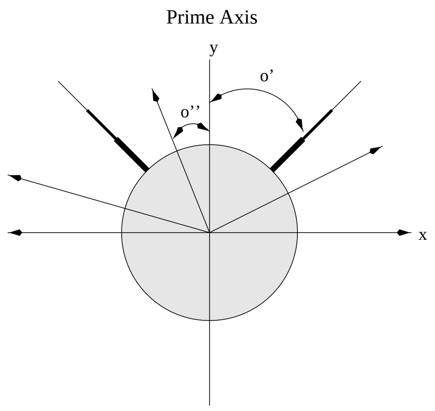

Figure 3: Spicule with Vector Recombination at t_n

2.2 Spicules and Tessellation

A tessellation surface is composed of a mosaic of cells and is typically found in geographic information systems [Chr97]. Tessellation surfaces are added to each other with surface algebra. Since a spicule is a graphical object, it can be tessellated in the same fashion a GIS grid can. There are several important benefits spicule tessellation allow. One of the benefits is the ability to construct spicules for systems based on a number of base case spicules. The ability to add these base spicules allows for a finer granularity then possible by modeling an entire system at once.

A second benefit allows for the creation of an attack spicule. To model the behavior of a bacteria attack, create a spicule of the feature vectors associated with user Bob's process activity. A bacteria attack can be characteristic of the rapid creation of child processes stemming from a single parent process. The spicule created would be very small, modeling a handful of system features. However, when Bob starts a bacteria attack the spicule represents only the characteristics of the system being effected. This creates a highly accurate model of the effects of Bob's bacteria attack. If a similar model exists for Alice, then by using tessellation to add the two spicules together, a composite spicule is created which contains a more refined model of the effects of a bacteria attack on the computer. By iterative tessellating and composing of spicules within a particular attack category a highly accurate attack spicule for modeling a particular attack is created.

Tessellation does not have to be constrained to a particular intrusion category. By the process above, it would be relatively simple to create an attack spicule for denial of service attacks. Once two attacks have been modeled, they can be tessellated into a representative spicule for both types of attacks. The ability to create tessellated spicules is the property which allows the creation of a model for system operation as well as a characteristic taxonomy for various categories of attacks.

Tessellation implies the ability to predict attacks by comparison and to determine what format attacks will look like on systems that have never been attacked. The following example demonstrates this characteristic. If system N has had a bacteria attack, it will have occurred during a certain time frame. In this case the pre-attack spicule was $N(t_m)$ and the post-attack spicule was $N(t_n)$. Since tessellation is also subtractive, the property $N(t_n) - N(t_m)$ produces a spicule reflecting only the essence of the attack. We will call this the attack form spicule. The attack form spicule contains strictly the additive vectors creating the post attack form, making the attack form portable and normalized.

This allows for the prediction of attacks on a computer. The attack form spicule can be tessellated to the system spicule of a computer; the result is a view of the computer as it would look after it has been attacked. With this knowledge it is then possible to monitor for changes in a system spicule which may indicate a transition to an attack form. With the proper granularity, it may be possible to potentially halt certain types of attacks before they occur.

2.3 The Singularity Model And Spicules

We have shown a spicule can model a single computer, but an equally important aspect of intrusion detection is the ability to detect attacks originating from other computers on a network, or other networks. When considering the relationship between computers, it is fairly simple to view this relationship in a geospatial (thus geometric) sense. The spatiality of this relationship can be expressed in logical terms. This essentially means computers can communicate with each other without concern as to their physical locations. But to share information, a computer must have some method of knowing of the existence of another computer. For communication on the Internet computers must have IP addresses, and when a computer has this information it can be said to $KNOW$ another computer. The strength of this $KNOW$s relationship could be viewed in terms of the number of packets per second with a particular IP associated with it. Patterns of intrusion occur on the pathways of these $KNOW$s relationships. Therefore it is appropriate to design models which incorporate this fact when attempting to design an IDS. Our model for a spicule does not contain the mechanisms necessary to deal with these relationships. A spicules geometry is relative to itself and mutually incompatible with visualizing the relationship between computers on the Internet.

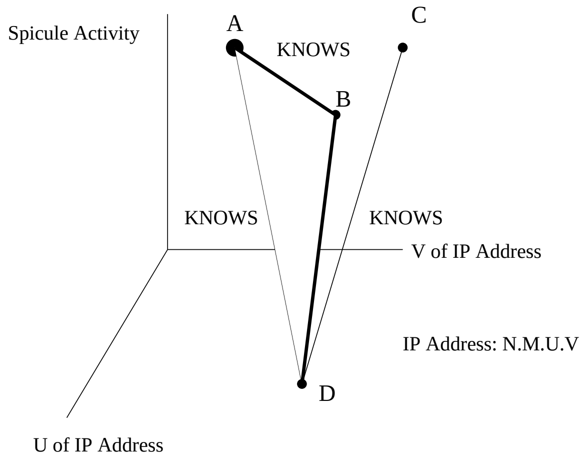

Figure 4: Sample Singularity Model at t_m

To solve this problem, the spicule for a computer on a network is collapsed into a singularity and incorporated into a singularity model. This singularity would be plotted in three dimensional space with its IP address on the x and z axis, coupled with a mathematical quantity derived from the spicules vector angles on the y axis. Initially this mathematical quantity represents the entire system activity of a spicule.

The singularity model is used in conjunction with spicules to detect intrusions. Within a singularity a spicule is undergoing a normal amount of flux caused by normal usage. However, when something strange occurs within the spicule it is represented by a corresponding change in its associated singularity. This allows for a singularity model to capture the relationship between spicules via the $KNOW$s relationship in terms of changes in the singularity. An example of this model is shown in Figure 4.

In Figure 4, for t_m the computer A has a $KNOW$s relationship with computer B. Since A is a file server and gets a lot of activity, this is modeled by its spicule singularity and can be seen by a difference in size.

3 Spicule and Singularity Model Example

As an example of how the spicule and singularity models work together we now consider how these models behave under various types of attack. We will visit the bacteria attack again for this example.

As stated previously, a bacteria attack occurs when a program consumes a large amount of a computer resources. This can be done by filling up disk space or by repeatedly forking child processes. These attacks can manifest itself in a variety of ways including decreased CPU throughput, increased CPU utilization, decreased free disk space, etc. Bacteria attacks are generally localized on a single computer, so a singularity model would notice a characteristic pattern of elevated activity for a single node, relative to neighbor nodes. For our example, a user will start forking child processes that perform CPU intensive activities as well as replicating themselves. The associated spicule will indicate changes in the system resources quadrant, particularly the CPU feature vectors, as well as changes in the user quadrant. Figure 5 shows our system before and after it is attacked.

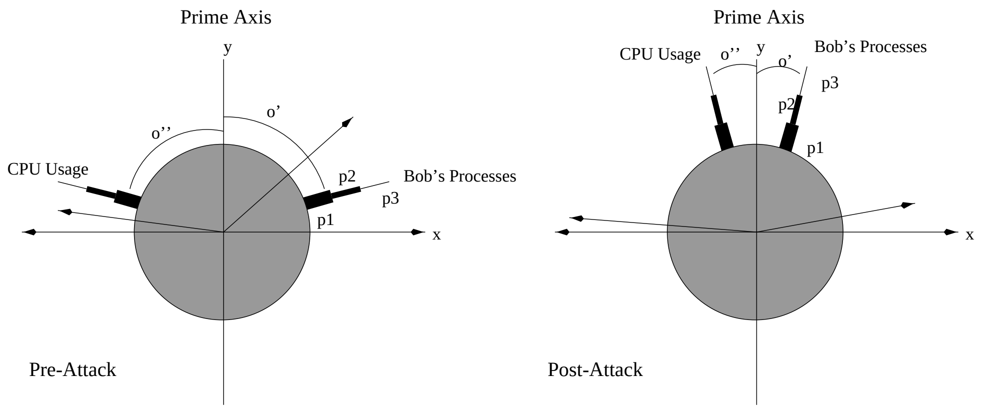

Figure 5: Pre/Post-Attacked Spicule for Computer A

While the attack is under way, CPU throughput will decrease as CPU utilization increases. As the attack forks processes, other processes in the user process quadrant will starve. In the system resources quadrant the three fastest changing feature vectors have added to create a recombination vector which shows that CPU time, the number of processes and percent CPU utilization have changed drastically. These changes can be seen in Figure 5.

For this example, the singularity model will have a characteristic pattern as well. Figure 6 is the singularity model before the attack.

Figure 7 shows the singularity model after the attack on System A. The normal $KNOW$s relationship computer A has with computer B has grown weaker because its CPU resources are being consumed by the bacteria, decreasing data throughput. The size of the singularity for computer As spicule is bigger as well, representing the increased activity.

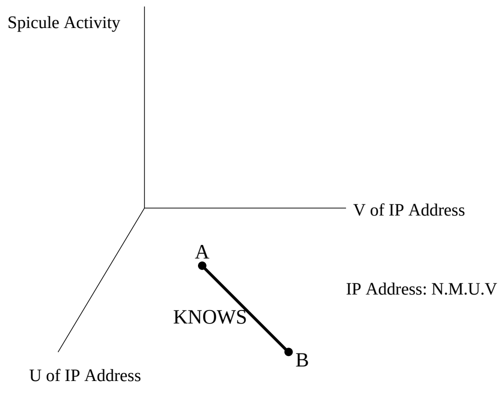

Figure 6: Pre-Bacteria Attacked Singularity Model Containing Computer A

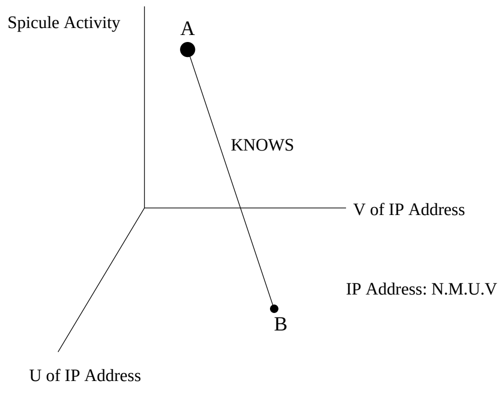

Figure 7: Post-Bacteria Attacked Singularity Model Containing Computer A

4 The Future of the Spicule Model

We are currently investigating the mappings between traditional intrusion detection approaches using statistics, expert systems with threshold detection, and signature recognition as mentioned in NIDES[VA94], DIDS[SB$^+$91] and Haystack/Stalker[Sma88] respectively. Initial testing indicates the spicule model can represent detection in ways similar to each of these models. We are also investigating the predictive and composite aspects spicules, to determine whether they are useful as a method of anticipating future attacks. Further refinement of the underlying spicule model is required before attack prediction becomes possible.

The spicule model is expected to provide an additional method of categorizing attacks and perhaps the specification of attacks. This should prove to be possible as feature vector development continues. Work is underway to create a set of base case spicules constructed from subsets of variables sensitive to certain types of attacks. Through tessellation of these base case spicules it should be possible to create spicule taxonomies which will be unique for certain types of attacks.

References

[Chr97] N. Chrisman. *Exploring Geographic Information Systems.* John Wiley and Sons, 1997.

[Gas83] P. Gasson. *Geometry of Spatial Forms.* Ellis Horwood Limited, 1983.

[HDL$^+$90] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A network security monitor. *Proceedings of the 1990 Symposium on Security and Privacy*, pages 296–304, May 1990.

[Ho97] Y. Ho. Partial order state transition analysis for an intrusion detection system. Master's thesis, University of Idaho, 1997.

[SB$^+$91] S. Snapp, J. Brentano, et al. DIDS (Distributed Intrusion Detection System)–Motivation, Architecture and An Early Prototype. *Proceedings of the 1991 National Computer Security Conference*, 1991.

[Sma88] S. Smaha. Haystack: An intrusion detection system. *Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference*, 1988.

[VA94] A. Valdes and D. Anderson. Statistical methods for computer usage anomaly detection using NIDES. *Conference on Rough Sets and Soft Computing*, November 1994.