A Visual Environment for Characterization of State Changes in Computer Systems

Gregory Vert, Sergiu Dascalu, Frederick Harris, Sermsak Buntha

Department of Computer Science and Engineering
University of Nevada, Reno, NV 89557, USA
E-mail: {gvert, dascalus, fredh, buntha}@cse.unr.edu

Abstract: Traditional state modeling techniques have several limitations. One of these is the reduced ability to model a large number of variables simultaneously. Another limitation is that understanding such models can be difficult as they are often represented as a series of graphs with a large number of transitions. In order to address the above issues, we propose a geometric approach to modeling state changes in computer systems. Geometric models are often descriptive of complex mathematical combinations and can be very useful for representing large quantities of intricate, interrelated data. Visualizations are powerful instruments of communication, therefore such geometric models can be processed very well by human beings. Visual state models can be utilized in a wide range of computer-related activities, from monitoring system operation to system security enhancement and data authentication. In this paper we describe a geometric model for visually representing various pieces of information pertaining to a computer system's security and operation aspects. Furthermore, we describe the main elements of a prototype software environment that supports such model and its related visualizations.

Keywords: spicule, computer state changes, visual model, visual environment.

1 Introduction

Operation and security of a computer system often requires processing large quantities of audit data, making it both computationally expensive and error prone [2]. To reduce computational requirements to a realistic level, administrative management of systems typically focus on a limited set of system and user attributes [3, 4]. Many types of sub systems in a computer need monitoring; among these are the network, processor, and file systems. Very little work has been done in this area. One such system FABS provides a user interface for monitoring anomalous accesses to a computers file system [5]. This system was conceived to deal with security threats and intrusion in a computer system.

Visualization of system information can be a powerful mechanism for comprehending complex and varied data. As an example, in the summer of 2000, the National Safe Skies Alliance awarded a project to the Applied Visualization Center at the University of Tennessee to develop a 3D computer tool to assist the US Federal Aviation Administration security group, in evaluating new equipment and procedures to improve airport checkpoint security. To date, several detection models have been developed for a few airports [6]. This effort demonstrates the broad range of areas that visualization of system operation can be applied to. However, a dynamic and comprehensive view of system operation (especially across a network of computers) is hard to encapsulate with just a few variables. Furthermore, system operation data has traditionally been presented in text form, which, given the typical volume, is difficult for administrators to process.

In order to answer the need to examine large quantities of data while alleviating the difficulty of comprehending such massive quantities, we propose enhancements to a prior geometric approach to modeling state changes in system data, based on using visual components. This model is called a Spicule [7]. Geometric models are often descriptive of intricate mathematical combinations and algebras and can be very useful for representing large quantities of complex, interrelated data. Visualizations are powerful instruments of communication, hence such geometric models can be efficiently processed by human beings. Visual state models can be utilized in a wide range of computer-related activities, from monitoring system operation to authentication of data and strengthening system security. In this paper we describe a geometric model based on vector mathematics intended to be used in computer system administration, security operations, and certain types of data authentication. Moreover, we present the main elements of a prototype software tool (environment) intended to support the model and allow the management and the displaying of its related visualizations.

The remainder of this paper is organized as follows: Section 2 reviews prior work where the

Spicule model was first introduced and describes the evolution and the proposed enhancements to the model, Section 3 presents excerpts of the software specification for the supporting visualization tool we propose, Section 4 provides details of the prototype GUI of the tool, and Section 5 finalizes the paper with directions of future work and conclusions.

2 Spicule Visualization

2.1 Previous Work

The concept of a Spicule was originally presented in [7]. In this work, the authors (led by the first author of this paper) presented a tool for intrusion detection that moved the data from standard text form to geometric primitives (circles in the 2D plane) with vectors emanating from the center.

The authors of [7] indicated that two things were needed to improve the performance of an intrusion detection system:

- A standard by which we can encode and represent complex computer systems in a unified way;
- A visual model for presenting information in a context where data may be represented and clearly seen as relationships between components.

In the above referenced work, the entire focus was on intrusion detection and categorization of attacks. The authors achieved their goals and presented a tool for IDS, but at that time they did not look completely into the power of the model they had presented.

2.2 Evolution of the Spicule Model

One important characteristic of a Spicule model is that it is not a static model. In other words, Spicule states change over time. Such a property is the basis for modeling state changes in a system. This allows for blending the visually intuitive with the mathematically quantifiable.

As we showed in the previous sub-section, the concept of a Spicule was originally developed for intrusion detection [7] but was not generalized to system state modeling. In the previous work, the Spicule was primarily described as a 2D object, while at this time we are expanding it more to a 3D sphere.

In addition to the central spheroid of the visualization, Spicules [7] incorporate *feature vectors*, each modeling a facet of system's activity. There are two types of feature vectors associated with a spicule. The first of this is referred to as a *tracking vector* and is utilized to represent system state data that can be mapped onto an interval ranging between 0% - 100%. As an example of a tracking one might consider CPU utilization, which is always at most 100%. A second type of vectors is referred to as *fixed vectors*. These vectors can not be mapped to a range but have the mathematical property of:

$$T_v = [\,0 \ldots \ \alpha\,]$$

Tracking and fixed vector behaviors differ. A tracking vector grows from the equator and is in a plane parallel to the ground. As the feature value increases, the tracking vector moves along a track on the surface of the spicule to the vertical axis, which represents 100% of the feature's value. Because this vector is a normalized vector, a mathematical signature may be computed from the angles of the tracking vectors.

In contrast, fixed vectors are located at the equator of the spicule, coplanar to the ground, and growing in magnitude parallel to the ground. For example, a fixed vector feature might be the representation of number of child processes spawned for a program [7]. One can see that there is no upper bound of 100% on this type of value and therefore fixed vectors can grow in magnitude infinitely. As such, fixed vectors provide magnitude of feature value information.

The three components of the spicule, namely security fitness, fixed vectors and the tracking vectors synthesize current approaches to visualization of state information and changes in state information for a system. Application to the determination of state changes can be performed in the following way. The angles and magnitudes of any vector can be pre-set for a normal state system and therefore thresholds can be established. Secondly, a set of tracking vectors with characteristic angles or fixed vectors with magnitudes for either normal or abnormal systems can be determined, providing a geometric signature. Comparing a system's model with this set of signatures lends itself to implementation of traditional state transition techniques.

In addition to the concept of classes of vectors representing state information about a computer system, the Spicule has the ability to model an infinite number of state variables at any given moment in time. This is a significant improvement over current state transition models that often model a transition based on the change in a single variable's value. For instance, if we model vector locations on the spicule to be discrete integer values, and the spicule is composed only of tracking vectors, the number of variables we can model is 360 and the number of states for the 360 variables is given as:

$$360 * 180 \rightarrow 64{,}800 \text{ state locations}$$

If we model tracking vectors on the real number system R the number of states and variables that can be modeled simultaneously becomes:

$$R * R \rightarrow \alpha$$

The extremely large modeling capacity of the Spicule dictates that it should have a very rapid method of communication with humans and thus we use a visualization because of the large amount of information content that can be conveyed to a user looking at a visualized state model. Because the spicule model is based on vector mathematics, a given spicule can be operated upon with vector algebra to eliminate unchanged information and display only information that has had state changes. The application of these properties ranges from intrusion detection, data authentication, system administration to creation of visual taxonomies of the characteristics of an attack on a computer system. A representation of a Spicule model is shown in Figure 1.

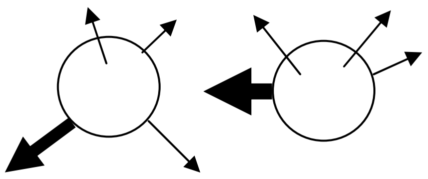

Figure 1: Representation of a 3D Spicule (top and side views)

In this figure the thick arrows are fixed vectors that grow in magnitude and thickness and the thin arrows are vectors that track from 0 degrees to 90 degrees on the spheroid.

2.3 Characterization of State Changes

As mentioned previously Spicules are expected to change states over time [7]. Therefore, we use the notation At_i to represent a Spicule's state model for node A at time t_i. The state of the spicule for the same node at t_0 is At_0. As such, we can have the notation of a set of state changes for a system represented as:

$$A = \{ At_0 \ldots At_i \ldots At_j \}$$

Additionally, regions on a Spicule can be partition into categories of related information. As an example, consider a Spicule with four information classifications regions in the upper hemisphere, grouped as system resources vectors, user processes vectors, system files vectors and system processes vectors. These quadrants may vary in number and type depending upon the information being represented. The appropriate set of feature vectors is dependent on those system characteristics, which are deemed most useful to a user for modeling purposes.

The process for updating vector state information should be automated. This means employing some sort of auditing tools to continuously collect data and send it to the Spicule's visualization software, where it can be dynamically rendered as a member of the set A. As an example, the process of state update and visualization might have the following steps:

The vertical component can be visualized as a percentage of the maximum value of 100%. For example, if the CPU usage was 20% at a given time, the vector representing CPU usage at that time would be visualized at 18 degrees along a track to the North pole of the spicule.

Vectors around the 360 degrees of the spheroid provide characteristics angles that can be measured for particular data feature vectors. Characteristic angles would constitute a geometric 'signature' for ongoing activity.

Vectors can be combined based on membership in the same category. The recombination property allows for geometric signature generation. For example, if the system resources category has the following members:

CPU usage	4% change/sec
CPU throughput	5% change/sec
port 25 traffic	1% change/sec
number writes	.05/sec
number reads	2% change/sec

then the recombination might involve only the three vectors with the highest rate of change over time or might combine (add) all vectors in the category.

3 Environment Specification

In this section, we briefly provide several functional and non-functional requirements [8, 9] of the current prototype of the visual system (environment) that supports the model. The functional requirements correspond to the use cases described in the next section. The functional requirements describe the operational capabilities intended to be provided to the system's users. The non-functional requirements are various types of constraints placed on the system, including implementation, performance, workload, usability, and maintenance constraints.

3.1 Functional Requirements

R1 The system shall allow adding and removing security characteristics to the visual model both before and during program running;
R2 The system shall represent system security characteristics using graphical vectors for visual modeling;
R3 The system shall display the color of the ball to indicate the security level in the system -- the color should be in a range from gray to red;
R4 The system shall provide a slider to allow a user go back and forth for watching history of the visual model. The slider should be associated with a color strip that indicates the historical color of the ball;
R5 The system shall show the time of the visual model that it is currently displayed. By default, it should show the current time. It should show time in the past when a user moves the slider to see the history of the visual model.
R6 The system shall allow a user to zoom in and out the visual model;
R7 The system shall allow a user to change the rotation speed of the visual model;
R8 The system shall rotate the spicule on vertical axis;
R9 The system shall provide a textual vector description when a user moves the mouse over the vector;
R10 The system shall show detailed information when a user clicks on a vector on the spicule in the visual model system.

3.2 Non-Functional Requirements

N1 The system shall support managing and displaying of at least 360 security characteristics in the visual model;
N2 The system shall keep the history of the visual model for at least 24 hours;
N3 The system shall provide detailed information on one vector at a time;
N4 The rotation speed shall be from zero to one round per second;
N5 In the case of a very long vector the system may be unable to display its entire graphical representation although the maximum zoom out is reached.

3.3 Use Case Diagram

In this section, the system behavior is described by use case diagram as shown in Figure 2. All use cases are designed for system administrators or whoever is in charge for system security. In the use case diagram, the person who is intended to use the system is called an "actor". Currently, our software prototype consists of one actor (who is both system administrator and user) and five main use cases, as the follows:

UC1 – Add/Remove security characteristics
UC2 – Watch previous spicule
UC3 – Zoom in/out
UC4 – Change rotational speed
UC5 – Acquire detail information of a vector

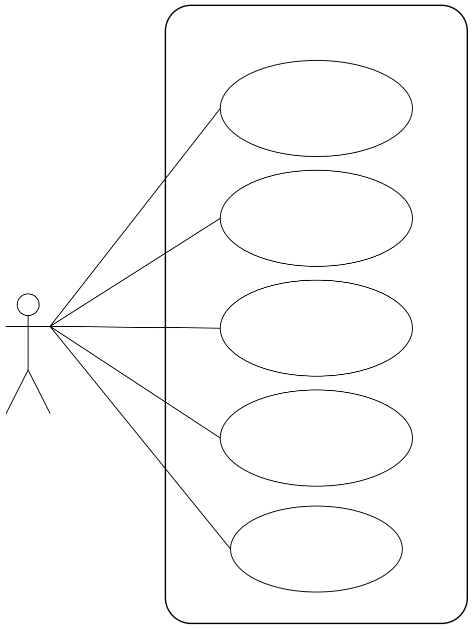

Figure 2: Use case diagram of the visual model system

The relationship between functional requirements and use cases is described with another software engineering tool, a requirements traceability matrix [8], depicted for our system in Table I. In our design, each use case completely implements one or more functional requirements of the system.

Table I: Requirements traceability matrix

Req	UC1	UC2	UC3	UC4	UC5
R1	•				
R2	•				
R3	•				
R4		•			
R5		•			
R6			•		
R7				•	
R8				•	
R9					•
R10					•

4 Prototype User Interface

Figure 3 shows the user interface of the proposed prototype system for spicule visualization. The largest portion of the environment's main window displays the visual model in the form of a spicule. The spicule consists of a ball (sphere) and a number of vectors pointing out from the ball. The color of the ball will change from gray to red when the system considers that there is a higher chance of being attacked. Each vector represents a characteristic item. The location and length of the vectors have already been described in the previous sections of this paper.

When the user moves the mouse over a vector, the brief deception value of the vector and the percentage value of its characteristic parameter changing will be displayed. Detailed information about the vector will be provided after the user clicks on the vector. On the bottom right corner of the visual model display there is a clock that shows the time associated with the spicule's activity.

Below the visual model area, the prototype visualization tool provides a function for watching previous changes in characteristics (historical evolution). The strip over the slider indicates the historical color of the ball in the last 24 hours. The slider is attached with five ticks and labels, respectively: 0h, -6h, -12h, -18h, and -24h. The marker "0h" indicates the current time and the others are related to the last 6, 12, 18, and 24 hours, respectively. The user is able to move the slider to observe the spicule during any selected periods of activity (e.g., of higher danger) or at any chosen points in time. In Figure 3, there has been a possibility of attack during the last 12 hours because the ball changed from gray to red. The clock on the bottom-right of the visual model displays the current time when the slider points at 0h (zero hour). Moving the slider to watch a previous state of the spicule makes the clock show the time associated with that previous state. This allows a user to investigate whether an attack occurred while the user did not monitor the system.

The spicule rotates to allow the user to observe it in its entirety (the sphere in all its 360 degrees). A user is able to change the rotation speed or even stop the rotation by moving the rotation slider n the bottom-left corner of the window. The slider on the bottom-right corner of the environment's main window is for zooming in and zooming out.

The visual environment for spicule management and display is currently under development and will likely be the subject of an extended description in one of our future publications.

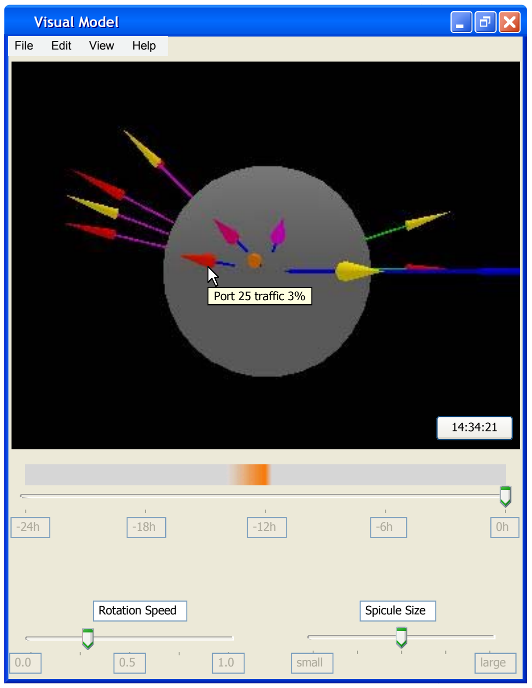

Figure 3: Proposed environment's GUI with a snapshot of the running program

5 Directions of Future Work and Conclusions

This initial research presents some initial theory and conceptual development of a system to model state changes in a computer system. It also presents some initial work on the extension of such a model to consider the relationship among Spicule systems located on distributed computers. Some of the benefits of such a model are: (i) the ability to model n dimensions of state data simultaneously; (ii)the ability to comprehend data rapidly in a visually intuitive fashion;

The application of such a model ranges from system administration to authentication of data with spatial components to administration of system and the construction of visual taxonomies of constituent Spicule models.

Furthermore, we have introduced the main elements of a prototype visualization system that supports the spicule model for monitoring computer state changes. This model will be further developed to become a useful tool in system administration with emphasis of security and safety concerns.

Much work remains to be done to validate and further test this model. We need to develop a better understanding of how to apply Spicule to some of the potential areas mentioned previously. Additionally, further work needs to be done on developing metrics that model relations among spicules. Also, data in a computer system is typically ambiguous. For example, it may sometimes mean one thing and at other times mean another. Finally, future work will concentrate on the application of fuzzy logic to the spicule model and what the semantics of such an application might mean.

References

[1] Chrisman, N., *Exploring Geographic Information Systems*, John Wiley & Sons, 1997.

[2] Heberlein, L., Dias G., Levitt, K., Mukherjee, B., Wood, J., Wolber, D., "A Network Security Monitor," *Proceedings of the Symposium on Security and Privacy,* p. 296-304, May 1990.

[3] Ho, Y., *Partial Order State Transition Analysis for an Intrusion Detection System*, Master's thesis, University of Idaho, 1997.

[4] Valdes A., and Anderson D., "Statistical Methods for Computer Usage Anomaly Detection Using NIDES," *Conference on Rough Sets and Soft Computing,* November 1994.

[5] Stanton, P.T. , Yurcik, W., and Brumbaugh, L., "FABS: File and Block Surveillance System for Determining Anomalous Disk Accesses, Systems," *Proceedings of the 6th Annual IEEE Systems Man and Cybernetics (SMC-2005), Information Assurance Workshop*, p. 207-214, June 2005.

[6] Koch, D.B. "3D Visualization to Support Airport Security Operations," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, issue 6, p. 23–28, June 2004.

[7] Vert, G., Frincke, D.A., and McConnell, J., "A Visual Mathematical Model for Intrusion Detection," *Proceedings of the 21st NISSC Conference,* Crystal City, Virginia, 1998.

[8] Arlow J. and Neustadt, I. *UML and the Unified Process: Practical Object-Oriented Analysis and Design*, Addison-Wesley, 2002.

[9] Sommerville, I. *Software Engineering*, 7th edition, Addison-Wesley, 2004.

.