A Theoretical Model for Detection of Advanced Persistent Threat in Networks and Systems Using a Finite Angular State Velocity Machine (FAST-VM)

Gregory Vert¹, Bilal Gonen², Jayson Brown³

¹²Department of Computer Information Systems, Texas A&M University – Central Texas, Killeen, TX, U.S.A.
²³Department of Computer Science, University of West Florida, Pensacola, FL, U.S.A.
¹greg.vert@ct.tamus.edu; ²bgonen@uwf.edu; ³jayson.brown@ct.tamus.edu

Received 25 November 2013; Accepted 15 January 2014; Published 15 May 2014
© 2014 Science and Engineering Publishing Company

Abstract

Intrusion detection systems have undergone numerous years of study and yet a great deal of problems remain; primarily a high percentage of false alarms and abysmal detection rates. A new type of threat has emerged that of Advanced Persistent Threat. This type of attack is known for being sophisticated and slow moving over a long period of time and is found in networked systems. Such threats may be detected by evaluation of large numbers of state variables describing complex system operation and state transitions over time. Analysis of such large numbers of variables is computationally inefficient especially if it is meant to be done in real time. The paper develops a completely new theoretical model that appears to be able to distill high order state variable data sets down to the essence of analytic changes in a system with APT operating. The model is based on the computationally efficient use of integer vectors. This approach has the capability to analyze threat over time, and has potential to detect, predict and classify new threat as being similar to threat already detected. The model presented is highly theoretical at this point with some initial prototype work demonstrated and some initial performance data.

Keywords

Network Security; High Order Data Analysis; Intrusion Detection Systems; APT; State Machines

Introduction

Computer security efforts are engaged in an asymmetric fight against an enemy comprised of thousands of both independent and interrelated actors on an incredible number of fronts across a grand surface area. Given the asymmetry, it is inefficient and simply not practical anymore for analysts to manually investigate each new attack. While this is true for the general case of known attacks, it is especially true for Advanced Persistent Threats (APTs). APTs are a cyber-crime category directed at business and political targets (Michael K. Daly, 2009). Traditional intrusion detection requires processing large quantities of audit data, making it both computationally expensive and error-prone (Dell SecureWorks, 2012) (Karen Scarfone, 2012) (Y. Kareev, 2009) (Z. Othman, 2010). Limitation of traditional IDS (Intrusion Detection System) techniques are as much a function of the ability of a human to process large amounts of information simultaneously as they are limitations of the techniques themselves (Neil MacDonald, 2010). New approaches to pattern recognition that simplify the process for human beings could be beneficial in better analysis of attack data.

Background

Much research has been done in the area of intrusion detection but little in its application to APT. In some related works, Shuo looked at the application of variable fuzzy sets to reduce the probability rates of false alarms and increasingly low detection of threats (Shuo-Liangxun, 2011). Variable sets were used to create a dynamic model capable of adapting to an ever evolving system state. By defining multi-characters within a state, they can deploy system mechanisms that monitor network activity adaptively to improve...
the implementation of a more reliable and more accurate set of detections. This in effect allows detection systems to be adaptive based on a multitude of personality traits found amongst complex systems rather than reactive to predefined security policies.

Vert et. al. looked at self organizing taxonomies that could detect and counter attack malware operations (G. Vert, 2012). In order to mitigate attacks on a system, methods were utilized to reduce sets of data containing attack signatures to only hardware specific state variables. By simply comparing previous states and how hardware state variables are affected over time, this research was able to build models that can respond to attacks based on fuzzy affinity matrices that can display natural trends of global system state variables over prolonged periods of data collection.

Work has also been conducted in the application of genetic algorithms (M. Hoque, 2012) to improve IDS operation. In this effort, GA was applied to detect previously undetected threat vectors, allowing more closely monitoring and conceptualization of network activity in a state system. Theories of evolution and ideas of adaptive characteristics based on these evolutionary state variables to dilute the complexity involved in threat detection were examined.

Very few approaches have sought to develop an analytic vector based algebra for detection. Some work however has been done looking at the role of visualization of security data (S. Eick, 1993) (Gensuo Han, 2012) (Bricken James, 1992) (Damballa, 2010). Previously, Vert et al. developed a conceptual mathematical model based on vector math and an analytic visual algebra for detecting intrusion attempts on computers. This approach was part of a larger project called “Spicule” (G. Vert, 1996) (R. Erbacher, 2002) (G. Vert, 1998). Spicule is a visualization technique that builds on this work and uses vectors to display the state of a system including its activity. This is a potentially powerful approach to intrusion detection because it uses vectors that offer themselves to various mathematical algebraic operations. Spicule can exist on many network hosts to aid in network security analysis.

This paper develops the theoretical model of a new concept referred to as a Finite State Angular Velocity Machine (FAST-VM) as a possible means to address APT on hosts in a network. It conjectures that APT could be detected and characterized by using very large data sets of network or host based state variables collected over long periods of time. The FAST-VM theoretical model has been developed to reduce high order state change data in networks and systems to the essence of what characterizes an APT presence in the system and to use a mathematical method that is computationally efficient and could analyze hundreds of thousands (>1,000,000) state variables in milliseconds.

**Advanced Persistent Threats**

By definition, an advanced, persistent threat, or APT, is a highly sophisticated networked entity, typical of organized groups that conduct hostile cyber-attacks against connected computers; whether on a local network or the internet (G. Vert, 2009). The intent of this type of threat may include one of the following (Dell Secure Works, 2012), i) gaining critical data found within government, financial or individual corporations, ii) maintaining access for future malicious intent, iii) manipulating data to disrupt the performance or operation of the target.

**Approach**

**Mathematical Foundations**

Integer based vector mathematics can be utilized to model state variables in a system. State variables are hardware and software attributes that change based on the systems operation; for instance CPU usage. State variables are indicators of the normal operation of a system and can assist in indicating the presence of abnormal operation or a threat.

The model proposed in this research is to use integer based vectors to model the state changes found in a system affected by APT presence. These vectors mitigate problems with the application of standard statistical methods for APT detection stemming from variation between hosts. Previous applications were extremely promising but focused on simpler exploits. Due to the subtle and sophisticated nature of APTs, there is no known upper bound on the number of state variable vectors that may be required for accurate detection. The underlying algebra has been shown experimentally to easily and efficiently handle thousands of state variable vectors without an appreciable degradation in execution time in large part due to being based on integer calculations. This allows for the reduction of high order state variable vector spaces to only the “needles in the haystack” which are indicative of APT presence.

Due to the persistent nature of an APT, vector modelling must also follow state variable changes over
a temporal range and across a geographic range as found in hosts on a network. Additionally, APTs are generally suspected to be mounted by sophisticated players. This necessitates the need to model a high order data set of state variables and their changes over time. The Finite Angular State Transition Velocity Machine (FAST-VM) has the capability to address these challenges. It can reduce the high order of state variable changes that have subtle changes in them over time to an easy to comprehend threat analysis that is the essence of APT’s effects on a systems state variable.

A few types of mathematical approaches could be utilized to potentially detect APT presence. One of the most likely would be statistical methods. There are a few reasons why this approach is not taken in the FAST-VM model. The first of these is that APT attacks are systemic and complicated and the second is that statistical methods start to fail when hundreds of thousands of variable need to be analyzed in real time over a temporal range.

APT’s are also expected to be detectable by way of monitoring a large number of networks and host state variables and analyze them simultaneously into a single aggregated picture for interpretation. The FAST-VM method has the capability to do this rapidly for any number of state variables at the same time. This could range from 10 to 1,000,000 state variables, all representing some aspect of a host’s operation in a network of hosts under APT attack. Performance is discussed and analyzed in later sections of this paper.

**Vector Mathematics Background**

Vectors have a variety of expressions usually denoted by a lower case letter. They can have magnitudes and point to locations in space indicating a particular value for a state variable, or they point to a fixed location and grow in magnitude to indicate changes in state variable values. The FAST-VM uses a combination of these types of variables. The vector based approach of the FAST-VM model is simple computationally efficient as discussed later and lends itself to an algebra that can detect state variable changes or predict what a state variable will look like if an APT has affected its value. This allows the vectors to:

(i) **Detect** change in a state variable if an APT has infected a system.

(ii) **Predict** what a state variable vector would look like if an APT is present and affects its value.

Vectors in the FAST-VM model are given as \([x, y, z] = v\) where \(v\) is a changing vector based point in space representing the value of the state variable that the vector is modelling. Using this model it is possible to have hundreds of thousands of vectors modelling very complicated system operation in a network over a given time period. For example given a state variable vector \(v\) whose value at time \(t\) is collected, and \(w\) for the same state variable collected at time \(t_i\), the operation of subtraction has the following result if the values of the vectors have not changed:

\[
w - v = y
\]

If \(y = 0\), no change to the state variable is detected suggesting no APT presence. This is referred as a **Zero** form. If \(y = 0\), the effect of APT presence on the state is indicated and is referred to as an observer form or an **Attack** form if the APT was previously detected. If \(w - v\) then \(y = 0\) indicating the effect of APT on a given state variable. The jitter part of the model that is beyond the scope of this paper does address the notion of being able to say \(w\) and \(v\) are slightly different but essentially the same over time so as to reduce the FPR and FNR rates.

The above property is binary and the basis for detection of change in state variables affected by APT’s over time. The vectors in the above example are referred to as the following:

- **v**: Normal form
- **w**: Change form – what has changed since \(v\) was sampled
- **y**: if \(y = 0\), \(w - v\) is referred to as an **Attack** form and indicates APT presence. If \(y = 0\) then no APT is present and \(w - v\) is referred to as a **Zero** form.

The algebra that predicts the effects of an unknown APT on what a state variable’s value may be given a previously detected APT is defined as the following:

- **v**: Normal form – state variable without APT present
- **z**: Attack form – previously detected class of APT effects on the state variable
- **p**: Predict form – APT effects on a state variable previously detected
- **o**: unknown APT affecting a given state variable

Note that \(v + z = p\). If an unknown APT is similar to a previously seen APT, then \(o - p = q\) where if \(q = 0\) indicates the presence of APT and is referred to as a
The usefulness of prediction is in the application of previously developed mitigation methods for APT signatures of previously known APTs and the similarity of an unknown to a known APT. This is part of continuing and future work to define and evaluate.

**Spicule Approach**

Analytic vector mathematics can be used to redefine the mathematics in a spatial extent (Damballa, 2010). This type of visual rendering is not diagrams or pictures; it has an analytic algebra that can be utilized to analyse data as discussed in the previous section (v, z, p, o, w, y). This is the concept behind the development of a data representation of high order state variable vector data called Spicule (G. Vert, 2012). Spicule does its analysis based on concepts presented in the previous section based on modelling variables, referred to as state variables that describe some aspect of a system’s or networks operation. It is possible to analyze up to tens of thousands of individual state variables and their changes to determine APT presence using the concepts previously presented. The analysis starts with population of state variable vectors around the radius of a Spicule in as small a degree increment as is required. Analysis for change (APT presence) is almost instantaneous using the fastest computational operation on a computer: integer addition and subtraction of vectors. Spicule’s mathematical model and underpinning is based on a vector calculus. Its algebraic visual model can do the following:

- **Detect (c)** changes to a system instantly by only visualizing what has changed in the system. This facilitates human interpretation of the significance of the change and its potential threat. It also lends to automatic response and classification of malware activity.

- **Predict (p)** what a system will look like under attack.

- **Identify (z)** the essence of how an attack changes a system.

- **Determine (Zero form to be discussed)** if the states of a system or network have changed or not. This is a similarity operator in the algebra.

The Spicule while mathematically analytic, does consider the need to have analysts involved in the analysis of large state variable data sets in an easy to understand fashion. The interface is simple and intuitive for humans to interpret as shown in the next section. It lends nicely to interpretation of events in a system facilitating human ways of reasoning about and interpreting a possible APT attack (e.g., “most likely APT,” “no APT,” and so on). In essence the needle in the haystack can be found easily using the vector approach coupled with human interpretation of the vector states previously discussed.

**Spicule Operation**

While the goal of this effort is not to produce a prototypical visualization system for APTs, it is self-evident from previous work that the visualization is a useful tool in the interpretation of large amounts of complicated data. The Spicule is displayed as a sphere with two types of state variable vectors. There can be almost an infinite number of these vectors representing thousands of state variables for a given host or network of hosts. The two types of vectors are defined as:

- **Fixed vectors** (vector G in figure 1) that represent state variables ranging from 0 to ∞; for example, the number of users that are logged on the system.

- **Tracking vectors** (vector B in figure 1) that range in value from {0,100}%; for example, CPU usage.

Each vector is located at a degree location around the equator of the Spicule ball and represents a state variable that is being monitored for change. In a simple case, with tracking vectors ranging from 0 to 90 degrees located 360 degrees around the equator, and the tip of each tracking vector indicating a state the system is in or state variable value, it is possible to model 32,400 unique states at any given moment in time and instantly analyze change between Spicules from two moments in time to see if malware is active using the Zero form in figure 5. Subdivision of degree locations for the vectors around the equator leads mathematically to an almost infinite number of states that could be modelled.

A Zero form, shown in Figure 5 (at the bottom as a round featureless ball), results when a Spicule at time t1 is subtracted from a Spicule at time t0 and no change has occurred in state variables being modelled by the tracking and fixed vectors. A Zero form indicates that no malware is in operation and the state of the system has not changed. Research beyond the scope of this paper is defining methods to say vectors are similar even if they do not have the exact same value. This phenomenon is referred to as “jitter” and
deals with false positive and negative rates.

![FIG. 1 SPICULE SHOWING SEVERAL PORTS ON A SYSTEM](image1)

The Spicule approach is to display system activity or state variables in the form of state variable vectors which are projected from the center of a sphere as in Figure 1. These vectors move or track as changes occur over time in a system. For example, a vector may represent CPU usage which can range from 0% to 100%. A CPU usage vector would normally start out at the equator to denote low CPU usage, but if the system found itself in the middle of a DOS (denial of service) attack, that same vector would be translated to pointing out of the northern pole to denote high CPU usage (near 100%).

For the initial development of a working Spicule prototype, we chose to test the concept by monitoring ports. While this prototypical test is not directly targeted at the realm of APTs specifically, it does serve to illustrate the early concept and so is included. Figure 1 is the vector display of a system monitoring a few state variables such as ports 22 (SSH, labelled A), 23 (Telnet, labelled B), 80 (HTTP, labelled C), 110 (POP3, labelled D), 137 (NetBIOS Name Service, labelled E), and 443 (HTTPS, labelled G). To test the prototype Backdoor SubSeven was used to simulate attack activity on specific ports. Backdoor SubSeven is a well-known Trojan. SubSeven works by opening an arbitrary port, specified by the attacker, but it most commonly attacks ports 1243, 6776, and 27374. Figure 2 shows the same system as before except that it is now under attack from SubSeven. The difference between these two Spicules is this new purple-tipped vector (labelled H) which has appeared suddenly with a great deal of traffic on an otherwise reserved port (1243).

![FIG. 2 SPICULE SHOWING A SYSTEM UNDER A SUBSEVEN ATTACK](image2)

Figure 1 is called a Normal form and Figure 2 is called a Change form, because something has changed in the vector states (the attack). The mathematics of calculating the Attack form and the relative reduction of data and interpretation of change is illustrated graphically in Figure 3. This shows the essence of the Backdoor SubSeven attack and its effects on the system.

**Spicule Mathematical Properties and Analytic Vector Algebra**

The Spicule model is comprised of six unique states: Normal form, Zero form, Change form, Attack form, Observe form, and Predict form. All of these are generated utilizing the previously discussed section on vector algebra and mathematics.

**Definitions:**

Normal form, N - a network systems state variables in normal ranges of operation

Change form, C - a systems state variables that have been changed

Attack form, A - the representation of only the state variables modelling malware effects on the system

Predict form, P - a representation of what a Normal form would look like given a specific malware operation

Zero form, Z - a featureless Spicule indicating that two forms used to create it are exactly the same

Observe form, O - the result of an algebraic operation, can be the same as a Change form or a Zero form
It is important to note the type of the form depends on the types of the forms used to calculate it and the algebraic operation used in the calculation. The algebra and calculations are presented in the following sections.

The Normal (N) form is the state in which the system is operating normally and not under attack. Opposite to this is the Change form (C), which is a representation of a system under attack. The Attack form (A) is a signature (or isolated) view of an attack in progress that is occurring inside the Change form. The Attack forms can be stored in a database for later reference, in which case they become Predict forms which are predictions of potentially future attacks. The Observe form is a state which may or may not be an attack signature. Through the vector calculus, an Observe form can be compared to a Predict form. Each one of these forms has a unique appearance and mathematical signature.

Most operations to produce the above forms are accomplished by adding two forms (their state variable vectors) together or subtracting one from another. The algebra is performed by iterating through the vectors of each Spicule and performing individual vector operations depending on the algebraic function being calculated. For example, in order to isolate an attack and produce an Attack form, simply subtract the Change form from the Normal form as follows:

\[ A = N - C \]  

where A is the Attack form of Spicule, N is the Normal form, and C is the Change form as shown in figure 3. The algorithm for this above process is:

FOR EACH \( \psi \) on the Spicule:

\[ A_{\psi} = N_{\psi} - C_{\psi} \]

END FOR.

In figure 3, one can see the essence of the attack's visual characteristics in the Attack form (A). Such forms can be potentially stored into a database as the Attack form of SubSeven or the family of malware that operates similar to SubSeven. Once they become stored and classified, they become a Predict form used in the future to identify the same or similar classes of attacks. Attack forms can be created from pre-classification of attack families for the major families of malware or APT. They are stored and used for identification of future attacks as Predict forms. They are subtracted from a Change form to classify an attack if a Zero form results from the algebra. The Attack form of Spicule is a classification of a type or family of attacks based on how they change the system state variables over time.

An Observe form may or may not represent an Attack form. It is generated by subtracting Spicules at different points in time to see if any change vectors appear. It can then be subtracted with a Predict form stored in a database to classify the family of attack that is occurring on the system.

A Change form is always Spicule at time \( t \) that a Normal form (at time \( t \)) is subtracted from to calculate the Observe form. To summarize, one can detect an attack by using the following equation:

\[ O = N - C \]  

where O is the Observe form, N is the Normal form, and C is the Change form. The major difference between equations (1) and (2) is that the latter is used to create an observe form, which is a possible Attack form, whereas the former is used when creating an Attack form only. The reasoning behind this is that (1) will be used to create a library of all attacks ever witnessed, and the result of (2) will be used to detect an attack underway against attacks stored in our library.

The Observe form is potentially what an attack would look like while it is underway. It is compared against
the stored known Attack forms to identify an attack. The method of performing this comparison is an algebraic subtraction as shown in the following:

\[ Z = A - O \]  \hspace{1cm} (3)

where Z is the Zero form, A is the Attack form, and O is the observe form. Essentially the Zero form is a Similarity operation. Figure 5 shows the actual Spicules applied to the Observe from figure 4.

A Predict form is meant to determine what a system might look like if a given attack from a family of malware is present on the system. It is way to watch for such an event if it occurs. The Predict form is created by the additive property of the algebra. It is calculated as the following:

\[ P = N + A \]  \hspace{1cm} (4)

This produces what we expect an attack or APT activity to look like if it occurs. The subtraction operation, the similarity operation creating the Zero form then identifies and confirms that the APT or attack malware is present via:

\[ Z = P - C \]  \hspace{1cm} (5)

If a Zero form exists then the attack has been identified, classified, and can be responded to as shown in figure 5. It is important to note that a Zero form occur with subtraction of one set of state variable vectors from another when they exactly match or jitter control has been applied. Figure 5, shows the Zero form on the right which means that the Predict or Attack forms match the current changes in the system, the Observe form. Note that similarity results in a featureless Spicule hence the name Zero form. This drastically simplifies and speeds the process of recognition by analyst. Mathematically interpretation can be also automated because the result of subtraction produces a \( Z = (0, 0, 0) \). The potential gain in identification time has the possibility to extend Spicule methodology to real-time visual and/or automated detection.

**FAST-VM, Finite Angular State Transition Velocity Machine Model Concept**

The next part of our research extends Spicules vector algebra into a state machine that can model and analyze thousands of state variables over time and space. Time and spatial would be the type of modelling FAST-VM would perform for a large complex network operating over a period of time. This is appropriate because the challenge in APT detection is the sophistication of the threats operation and the stealth over time. Watching for a state variable change at some point in time is not sufficient. It is conjectured that there is a need to watch a very large number of state variables and then distill the essence of the APT's activities using the sort of high order data set reduction previously shown in the Spicule approach. If an attacker has an effect on a machine, the attacker will also change its state in subtle ways. These ways are not easily predictable unless the Spicule model is adapted to look at the velocity (rate) of state changes over time. This is the essence of the FAST-VM concept.

FAST-VM builds on Spicules algebraic analysis capabilities. It models a signature of Spicules as they transition from one state form for a portion of an attack to another state form as shown in Figure 6. The FAST-VM model transitions to new states which are measured by their state transition velocities. Velocity still a part of future research but informally is defined to be the rate of change in thousands of vectors over time to a new Spicule state. FAST-VM signatures or branches can generate in any direction in 3D space. Considering that a branch of a FAST-VM transition as shown in figure 6 could be located at every single degree location from an initial Spicule state, it is possible to model potentially huge amounts of data and variations of malware or APT operation. When an APT starts to generate a FAST-VM signature branch it can be compared to existing branches stored in a database to determine if it is similar to previous
patterns of activity. This comparison can allow a system to classify known and similar types of attacks in a computationally efficient manner because it is based on integer vector data.

The FAST-VM models high order data spaces of state variables as would be found on a large network that could be affected by an APT and reduces them to the essence of an APT’s effect on state variables as it operates over time. It has the ability to model thousands of state variable vectors and then perform the following key functions from the Spicule model:

- **Find** the needles in the haystack representing the APTs’ effects on the network through detection of changes in high order state variable data sets, (3, 2) the Observe form.
- **Detect** previously unknown APTs (1), the Attack form.
- **Classify** state activity changes of unknown APTs as similar to known APTs allowing mitigation methods for a known APT to be applied to an unknown APT once the FAST-VM has detected its presence, (5) the Zero form.
- **Predict** what a category of an APT’s attack on a system might look like so that it can be monitored for presence, (4) the Predict Form.

**FAST-VM Operation**

The FAST-VM model consists of a series of Spicules as they transition over time that can be combined into N-dimension state transitions or an elaborate graph. Each transition has a velocity which is still being defined. The velocity is defined by the rate of change in Spicule Change forms over time and possibly an attribute of probabilistic confidence that denotes the transitioned to state as a recognized state. In each state of the graph created by transitions to a new state, the Spicule algebra is applied, evaluating Change forms, and applying Predict form analysis. If a series of Predict forms match what is observed, then the APT can be classified and identified. The model of FAST-VM's operation is shown in Figure 6. In this example, one can see the following characteristics:

- Spicules representing state changes in state variable at various points in time;
- a velocity equation \(|h|\) (magnitude of the transition) that describes the transition speed from \(t_0\) to \(t_1\);
- a cumulative Attack form describing the attack signature for APT summed over time at \(t_1\);
- a Bayesian probability \(P\) based on confidence that the attack signatures are known to be part of the transition attack profile for malware, where \(M_x\) is malware \(x\) and \(P_n\) is the probability of having a known attack Spicule form at \(T_n\). \(P\) is thought to deal with the issue of jitter, in that for a given malware family, Spicule Attack forms at any given point in time should be similar but may not be exactly the same.

Figure 6 shows the operational aspects of FAST-VM with the Spicule algebra calculating change forms where:

\[T_n: \text{time slice}\]

\[|h|: \text{velocity as a function of magnitude of transition to a new Change form}\]

\[P_M: P_{t0}, P_{t0}, P_{t0}: \text{probability of malware APTx given probability time slices model components of unknown system operation, or known/similar APTx state transitions.}\]

The above diagram shows a FAST-VM creating a signature trail as an APT is being analyzed at each step in the process. This trail considers the rate of change to a new state over time and the high order of state variables that can be evaluated and reduced to the essence of the attack (Attack form or Change Forms) as each transition. Of note, only Change Forms are being...
modelled in figure 6. In reality each Spicule could have thousands of state variables that have not changed and thus are eliminated in the vector algebraic Spicule analysis.

**Performance Testing**

Finally, one of the key features of FAST-VM is its ability to model and analyze large amounts of data. This should be done with as low computational overhead as possible. The vector operations in the algebra were modelled in a simulation using Visual Studio 2008 on an Intel laptop with fours cores running Windows 7. The results are shown in the following tables

<table>
<thead>
<tr>
<th>TABLE I DIFFERENCING CALCULATIONS (−)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Vectors Evaluated (Integer)</td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>1,000,000</td>
</tr>
<tr>
<td>10,000,000</td>
</tr>
<tr>
<td>100,000,000</td>
</tr>
<tr>
<td>1,000,000,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II ADDITIVE CALCULATIONS (+)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Vectors Evaluated (Integer)</td>
</tr>
<tr>
<td>1,000</td>
</tr>
<tr>
<td>100,000</td>
</tr>
<tr>
<td>1,000,000</td>
</tr>
<tr>
<td>10,000,000</td>
</tr>
<tr>
<td>100,000,000</td>
</tr>
</tbody>
</table>

The resultant calculations suggest that FAST-VM could analyze and model large amounts of data across a complicated and active network with relatively low computational overhead. This will be the subject of future evaluation.

**Conclusions and Future Research**

Detection of sophisticated slow moving threats such as APT requires the ability to model large amounts of data and reduce it rapidly to the essence of what has changed in system state over time. Analysis then has to have the capability to create signatures that can be stored for normal state transition and compared against to predict future unknown but similar types of attacks. FAST-VM and its vector based analytic algebra now provide a potential powerful model for detection of advanced threats. As shown in the preliminary performance results FAST-VM can computationally efficiently analyze large amounts of data and can scale to network detection of hundreds of thousands of network states. In this theoretical model APT was chosen as a vehicle to demonstrate the FAST-VM model and how its theoretical underpinnings operate. There are many research areas still to be investigated to develop the model further.

The first of these areas involves research that centers on the isolation of state changes indicative of APT presence. There is also a degree of classification work that needs to be completed that attempts to define and quantify broad categories of APTs that share predictive characteristics. Another future research task is involved in the refinement of techniques that predict the expected state of a system; that is, there is no attack and things are operating normally. As this is intended to be prototypical and demonstrative, it is expected that, to a degree, our prediction will be tailored to a demonstration prototype system initially. Included in this is the identification of a large collection of state variables describing host and network operation from ongoing and past research.

Research needs to be further conducted to evaluate the adaptive jitter methods to deal with vectors that do not have exactly the same locations but are essentially the same. This phenomenon is referred to as jitter. A theoretical model is in development to deal with reduction of false positives and false positives.
and developed for networks of FAST-VMs. It is thought that cooperative colonies of FAST-VM's around a large network might be a promising approach to autonomous threat detection and perhaps mitigation.

REFERENCES

Dell SecureWorks, "Anatomy of an Advanced Persistent Threat (APT)". Retrieved 2012-05-21


Gregory Vert earned a PhD from the University of Idaho. He holds a CISSP and CCENT certification and is the founder and Coordinator of the Security Education Research and Training (SERT) lab at Texas A&M Central Texas. He completed his post-doc at LSU in the Computer Science department in 2009 and was appointed to the Center for Secure Cyberspace.

He has authored 70+ papers with most of them in the area of security. He also defined and published a book recently on the topic of Contextual Processing and contextually based security.

Dr. Vert is a member of ACM and IEEE and was the founder of these organizations at Texas A&M.

Bilal Gonen was born in Istanbul, Turkey. He received a B.E. in Computer Engineering in Turkey in 2002. He received his M.S. in Computer Science from the University of Georgia, U.S.A. in 2006. He received his Ph.D. in Computer Science from University of Nevada, Reno, in Nevada, U.S.A. in 2011.

He is currently an Assistant Professor in Computer Science department at the University of West Florida, in Florida, U.S.A. After completing his Ph.D. he worked as a Term Assistant Professor at University of Alaska, Anchorage (Alaska, U.S.A.). His research interests include Network
Science, Social Network Analysis, Semantic Web, Computer Networks, and Bioinformatics.

Dr. Gonen is a member of ACM, IEEE and Society for Science & the Public (SSP).

Jayson G. Brown was born in Abilene, Texas USA. He received his Master of Science in 2013 from Texas A&M University – Central Texas in the major of Information Systems as a Distinguished Student. His previous undergraduate background involves Computer Information Systems in Software Engineering with Summa Cum Laude honors.

He has conference proceedings relating to knowledge transfer and software methodologies. He is in the midst of finding a PhD program at the current time, but is maintaining employment as a Graduate Assistant with the CIS department. His personal research interests include software evolution, biometrics, AI, machine learning or other similar fields.

Jayson is a member of ACM, AMCIS and various honor societies including Delta Mu Delta.