

Chapter 3

A Technology for Detection of Advanced Persistent Threat in Networks and Systems Using a Finite Angular State Velocity Machine and Vector Mathematics

Gregory Vert, Ann Leslie Claesson-Vert, Jesse Roberts, and Erica Bott

3.1 Identification and Significance of the Problem or Opportunity

3.1.1 Introduction

Computers are an integral part of our society. 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 [1]. Given the asymmetry, it is inefficient and impractical for analysts to manually investigate each new attack [2]. While this is true for the general case of known attacks, it is especially true for Advanced Persistent Threats (APTs). Advanced Persistent Threats are cyber-crimes that occur when an unauthorized person accesses a network and remains for a prolonged time period to steal information as opposed to compromise the organization itself [2, 3]. APTs are used to target financial institutions, military defense, aerospace, healthcare, manufacturing industries, technologies, public utilities, or political entities [2–5]. In discussing potential automation of parts of the process or the entirety of it all, it behooves us to observe the process first. When a new attack occurs, detection is the first step. Once detected, the offending code and its effects are isolated from background noise and studied. That is, its function must be determined, as well as potential relationships,

G. Vert (✉) • J. Roberts • E. Bott

College of Security and Intelligence, Embry-Riddle Aeronautical University, Prescott, AZ, USA
e-mail: gvert@erau.edu; roberj49@my.erau.edu; botte@my.erau.edu

A.L. Claesson-Vert

School of Nursing, College of Health and Human Services, Northern Arizona University, Flagstaff, AZ, USA

e-mail: Ann.Claesson@nau.edu

if any, to preexisting attacks. While it may present with a previously undiscovered and unique signature, that signature may still bear some semblance to prior attacks. Function refers to the goal of the attack; e.g., disruption of the service. This relationship can both be familial, as in a new variant of an existing exploit, or typical as in a broad type (i.e., category) of attack. Lastly, the design and implementation of preventative and ameliorative efforts to remove the existing infection are crucial. It is in these prevention and amelioration tasks that the previously determined factors such as type, family, function, and code become important.

While automation of the entirety of the process is attractive, this technology focuses on the automation of the identification and classification stages of the process [2]. The major focus is on detection, though a certain degree of type classification is possible and desirable as well. Additionally, this classification will assist both external efforts in amelioration as well as serve to reduce false positive rates. This not only indicates unauthorized activity, but it matches patterns of behavior associated with a given broad category of Advanced Persistent Threat.

3.1.2 Background and Significance

Research has identified limitations of traditional Intrusion Detection Systems (IDS) in the areas of human error, cost, and high error-rates due to large volumes of data being processed simultaneously. Traditional intrusion detection requires processing large quantities of audit data, making it both computationally expensive and error-prone [2, 5–7]. 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 [4–7]. Machines currently have a limited ability to recognize unusual states or attack states. Humans are more effective at properly recognizing anomalies, yet have limited ability to consistently and effectively sift through large amounts of data [4]. These findings support the need for alternative automated approaches to pattern recognition for better analysis of real or perceived APT attack [2, 3, 5–9]. New approaches to pattern recognition that simplify the process for human beings, such as FAST-VM, are beneficial in better analysis of attack data.

Data visualization and visual algebra is a potentially useful method for dealing with the limitations of tabular log-based data, due to the amount of information it can organize and present [10–16]. Additional research stressed methods to define and mediate APTs, malware, and their reduction of false positive rates. Self-organizing taxonomies that can detect and counter attack malware operations were examined by Vert et al. [17], application of variable fuzzy sets to reduce false alarms by Shuo et al. [18], and the application of genetic algorithms by Hoque et al. [19]. Another major issue facing traditional IDS techniques is they are either signature based and fail to detect brand new attacks, or are anomaly based and are prone to false positive rates far more than what human operators can handle. Since IDS operators are already overwhelmed by the volume of data, an excess of false positives would be problematic.

Prior research by Vert et al. resulted in the development of a conceptual mathematical model based on vector math and analytic visual algebra for detecting intrusion attempts on computers based on the concept of identification of normal versus abnormal fluctuation patterns of known vectors. This approach was part of a larger project called “Spicule” [2, 20]. Spicule is a visualization technique that builds on this work and uses vectors to display system activity. A Spicule is an easy way for a human to visualize a large amount of data in a short span of time. This is a potentially powerful approach to intrusion detection because it uses vectors that offer themselves to various mathematical operations. Using mathematical properties allows the same data set to take on different semantics. Furthermore, detection and classification become automated through the inherent algebra of Spicule [20]. Although not the primary focus of this work, it was anticipated that the visualization capabilities greatly assist the developmental efforts and may provide a future area for feature expansion. The focus of this work is on the utilization of extremely mathematically efficient algorithms as a detection mechanism.

Erbacher et al. built upon existing work by Vert et al. [20] and the Spicule concept to develop a visualization system that monitored a large-scale computer network [14]. Their system monitors computer activity using several glyphs in conjunction with time. Chandran, Hrudya, and Poornachandran referenced the FAST-VM model in their 2015 research on efficient classification roles for APT detection [21]. This research is an example of the broader applicability of this area of research.

3.1.3 Problems and Opportunities

The area of non-signature-based intrusion detection is a challenging, yet rich area for investigation. While APTs have greater access to novel and unique attacks, they do not have exclusive access. Rather, zero day exploits are a constant factor in computer security. As a result, all security companies are constantly scrambling to provide signatures to emergent attacks to protect their customers.

Any techniques capable of addressing APTs can be retooled to address general computer security. These techniques are practically guaranteed to be novel; as traditional signature-based detection mechanisms are not reactive enough to catch the novelty of APTs. To put it simply, research into APT detection and prevention systems has the potential to be game changing as evidenced by the existing research areas [1, 2, 4–6, 8, 14, 20].

This does not overshadow the challenge of addressing APTs. It is not sufficient to look for a signature of a specific piece of code, or exploit alone [1, 2]. A true understanding of the state of the machines is needed. If an attacker influences a machine, the attacker may change its state in subtle ways that are not easily predictable. It might be possible to classify attacks or types of state changes, however. This is hindered further by the sheer dimensionality of the data when the state of a machine is considered. One cannot simply rely on a human log-file type

analysis to infer these state relationships; better detection mechanisms are necessary [2, 3, 5–9].

Another potential approach is that of simply trying to check for anomalous states—states that the machine does not normally enter. Unfortunately, the set of attacks seem to be a miniscule subset of the set of all anomalous states if current research is any indication. This leads to false positive rates that are simply intractable. Therefore, while it is important to try to predict expected “normal” states, it alone is insufficient.

Truthfully, it is likely any successful approach must take some combination of approaches, both predicting expected machine states and attempting to understand the interrelation between state change and attack [22–24]. This process is used to crosscheck anomalous states with states that indicate an attack. While humans alone cannot be relied upon to make this determination, there is certainly room to leverage human intuition in this process. Potential approaches vary, but include both computational techniques and visualization techniques. The implications of even partial functional success (as opposed to purely theoretical success) are many fold, and potentially ripple through many domains.

3.2 Concept

Our concept is to use vectors to model state changes affected by APTs. These vectors mitigate problems with the application of standard statistical methods for APT detection stemming from variation between hosts. Additionally, the algebra of the vectors, once decomposed to its base elements, is intuitive and easy to analyze for the security analyst or researcher [24]. This is of great importance currently in the developmental and research phases. Later this capability could extend into a general visualization to aid detection at a production level. There has been exploration conceptually and experimentally of the vector math and visualization. However, the domain of APTs is a new application area. Previous applications were extremely promising, but focused on simpler exploits.

There is no known upper bound on the number of state variable vectors that may be required for accurate detection. This is due to the subtle and sophisticated nature of APTs. The underlying algebra demonstrates 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 upon 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. Anything less efficient would be unacceptably computationally expensive for practical detection of threats.

We utilize expected state prediction to reduce false positives rates (FPRs). We refer to this capability as Jitter, which is adaptive over time. It uses sweep region statistical analysis and adaptive adjustment. It can also utilize Bayesian probabilities based on previous locations for a state variable vector. This allows the change from

algorithm to remove vectors that are jittering and known versus truly new APT effects on vectors. Jitter is subject to future developments.

FAST-VM unifies the three major areas of IDS (anomaly, misuse, and specification) into a single model. FAST-VM is a signature of Spicules, the experimental visual form of the vector math, as they transition to new states, which can be measured by their velocity. Velocity is 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. When an APT starts to generate a FAST-VM signature branch, it can be compared to existing branches. This determines if it is similar to previous patterns of misuse. This comparison allows the system to mitigate APTs during instantiation, prior to activation.

3.2.1 Technical Objectives

Our goal was to create a preliminary prototype that models high order data spaces of state variables that could be affected by APT and reduce them to the essence of the operations of that APT over time. This is the Spicule vector mathematics model that integrates into the Finite Angular State Velocity Transition Machine (FAST-VM). The FAST-VM performs the key functions found in Table 3.1.

A real-time Spicule has been implemented as a prototype. The algebra has solid mathematical underpinnings as well as circumvents problems with application of statistical analysis methods. Further discussion of this appears later in this chapter.

The goals of this technology are:

1. Identify categories of APT attack and activities and collect or develop software tools to simulate APT activity in a networked environment.
2. Using current research, identify a large collection of state variables describing host and network operation. Implement and test FAST-VM against several categories of APT. Evaluate and refine its model including identification of state variables that are most sensitive and diagnostic of various categories of APT threat.
3. Implement and evaluate the adaptive Jitter methods previously discussed to measure and fine tune the FRR and FPR rates.

Table 3.1 Fast-VM capabilities

	Capability
1	<i>Find</i> “the needles in the haystack” representing the APT effects on the system
2	<i>Detect</i> a previously unknown APT
3	<i>Classify</i> state activity changes of unknown APTs as similar to known APTs
4	<i>Predict</i> what this category of APT attack on a system might look like so that it can be monitored for presence

3.3 Implementation

3.3.1 Overview

The broad strokes of the concept were summarized in the preceding concept section. The implementation for FAST-VM revisits each of those topics in detail before laying out a plan and series of tasks for moving forward. Integer-based vector mathematics is central to this concept and the approach and its justifications are discussed in detail. First vector mathematics is discussed, then the mechanisms used to represent threat on a system. This leads to a consideration of time effects, and how the complexity of APTs is captured through time. Finally, the mechanisms for counteracting the problems of FPRs and FRRs, which tend to plague non-invariant-based approaches such as anomaly detection, are defined.

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. APTs require a high degree of stealth over a prolonged duration of operation to be successful.

Threats on a system or network can be detected as changes in state variables if the correct variables are modeled. Due to the persistence nature of an APT, vector modeling also needs to follow state variable changes in a temporal range. Additionally, APT attacks are generally mounted by sophisticated actors. This necessitates the need to model a high order data set of state variables and their changes over time. The 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.

FAST-VM is also integer based. Integer mathematics is one of the fastest ALU operations on most computers. In some past tests utilizing FAST-VM concepts to generate authentication signatures and comparing that to cryptographic methods, FAST-VM generated vector signatures approximately fifty times faster [2]. Basing FAST-VM model on this approach, it is possible that this technology could run in real time or near real time to detect APT presence.

Finally, it is important to address false positives and false negatives in any APT system. State variables are going to have normal ranges of operation that will change over time. A mechanism called adaptive jitter addresses this. The model allows a system to dynamically adapt its jitter function for the location of state variable vectors such that the false positive and false negative rates can automatically be reduced and perhaps eliminated. The user of FAST-VM can change jitter values as they see fit so they get the desired false positive rate.

3.3.2 Vector Mathematics Versus Other Methods

A few types of mathematical approaches could be utilized to potentially detect APT presence. Most likely would be statistical methods, producing state variable

averages and standard deviations. 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 a systemic and complicated attack. Statistical methods for state variables on a single host are fine for detection. A problem arises, however, when comparing, interpreting, and analyzing state variable statistics across multiple platforms. This is far too computationally intensive. Additionally, standard deviations, which might indicate APT presence, have different values among different hosts that APT may be in operation on. This leads to the question of how to compare systems and conduct a meaningful analysis among all systems. The binary properties of the vector approach are meant to display no vectors or few vectors if a system does not have APT present and large numbers of vectors if it is present. As noted below, this makes interpretation by an analyst much easier.

APT is also detected by monitoring many state variables and analyzing 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 10000 variables all representing some aspect of a host's operation in a network under APT attack. The analysis is done in a human intuitive fashion which makes training people to use the system easy. Similar sorts of capability using statistical methods get expensive computationally and are complicated to interpret.

There are other arguments that can be made for a vector-based approach, but in the final evaluation, a human analyst is required to interpret the data to determine the presence of an APT in a specified system of computers. Humans are highly skilled at fuzzy thinking. The vector method allows for an extremely intuitive method for analysis of the APT data from a system of hosts and the interpretation of such data thus aiding the human analyst. Subsequent sections further discuss this approach.

3.3.3 *Vector Mathematics Background*

Vectors have a variety of expressions usually denoted by a lower-case letter. They have magnitudes and point to locations in space indicating a precise 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 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:

1. Detect change in a state variable if an APT has started operation on a system.
2. Predict what a state variable vector would look like if an APT is present and affects its value.

Looking at (1) first, given state variable vector v whose value at time t_0 is collected and w for the same state variable collected at time t_{0+1} .

The operation of subtraction has the following result if the values of the vectors have not changed

$$w-v = y$$

where v is the *Normal Form vector*, and w is the *Change Form vector*, what has changed since v was sampled.

If $y = 0$, no change to the state variable has been detected, suggesting no APT presence. This is referred to as a *Zero Form*.

If $y \neq 0$, the effect of APT presence on the state is indicated and is referred to as an *Observe Form*. If this specific state change has been detected previously and associated with a known attack, the change is referred to as an *Attack Form*.

If $w \neq v$, then $y \neq 0$, indicating the effect of APT on a given state variable. The *jitter* part of the model does address the notion of being able to say w and v are slightly different but essentially the same over time to reduce false readings.

Considering prediction of the effects of an unknown APT on what a state variable's value might be given a similar category of APT that has previously been detected, its algebra is defined as follows:

v - *Normal form*, state variable without APT present

z - Previously detected class of APT effects on the state variable, referred to as the *Attack Form*

p - *Predicted form* of an APT effects on a state variable previously detected, referred to as a predict form

o - Unknown APT affecting a given state variable.

$$v + z = p$$

Equation for predict form

If an unknown APT is similar to a previously seen APT, then:

$$o-p = q$$

q - If zero indicates the presence of APT, referred to as a *Zero Form*. If not equal to zero, indicates that the APT is new and previously unknown but has now been detected.

The usefulness of prediction is in application of previously developed mitigation methods rapidly versus having to develop new mitigation methods for a previously unknown APT. The FAST-VM model also allows rapid classification of an attack using algebra and logic like the above predict form calculation.

3.3.4 Previous Work and Example Approach

The vector mathematics and algebra previously presented can be extended to model a state variable environment consisting of thousands of variables that could be utilized to detect the subtle changes APT might have on a system of computers. Because of the binary property of differencing (if a future vector is the same as a past vector the resultant is zero) and the application of jitter control previously discussed, a model containing thousands of state variable vectors from the past and future can be differenced to reduce high order data to the essence of exactly what has changed. This can cull out the essence of APT effects on state variable and thus can be analyzed to determine for potential presence of APT as presented in the following sections about Spicule and FAST-VM.

Analytic visual mathematics can be used to redefine mathematics spatially [2]. This type of visual rendering is not diagrams or pictures, it has an algebra that can be utilized to analyze data. This is the concept behind the development of a 3D data representation of high order state variable vector data Spicule. Spicule does this by modeling variables describing a system's operation. It is possible to analyze up to tens of thousands of individual state variables and their change to determine APT presence. This is done by population of state variable vectors around the radius of a Spicule in as small a degree increment as required. Analysis for change, and thus APT presence, is almost instantaneous using the fastest computational operation on a computer, integer addition and subtraction of vectors data. Spicule's mathematical model and underpinning is based on a vector calculus. Its algebraic visual model can do the following:

1. *Detect changes* to a system instantly by only visualizing what has changed in the system (this form of Spicule is referred to as the *Change Form*). 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.
2. *Predict* what a system will look like under attack (referred to as the *Predict Form*).
3. *Identify the essence* of how an attack changes a system (referred to as the *Attack Form*).
4. Determine if the states of a system have changed or not changed (referred to as the *Zero Form* or the *Ball Form*).

The Spicule interface is simple and intuitive for humans to interpret and requires very little training. It lends nicely to interpretation of events in a system facilitating human/fuzzy ways of reasoning and interpreting a possible APT attack such as "most likely APT," "no APT," "sort of similar to previous APT," or types of change analysis. The *Change Form* finds the "needles in the haystack" of a high order state variable data space and presents that alone for analysis of APT presence.

3.3.5 Visualization Work: Spicule

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 developmental and research phases. In short, it is a feature that can be later developed to enhance any resultant product once said product has been proven.

The Spicule is visualized as a sphere with two types of state variable vectors (Fig. 3.1). There can be an infinite number of these vectors representing thousands of state variable for a given host, or network of hosts. The two types of vectors are defined as:

1. Fixed vectors (green) that represent state variables ranging from 0 to infinity; for example, the number of users that are logged into the system.
2. Tracking vectors (blue) that range in value from 0 to 100% and track scalar state variables; for example, CPU usage.

Each vector is located at a degree location around the equator of the Spicule ball. Each vector represents a state variable that is being monitored for change. In a simple case, with tracking vectors ranging from 0 to 90° located 360° around the equator, and the tip of each tracking vector indicating a state the system is in, it is possible to model 32,400 (90×360) unique states at any given moment in time. This makes it possible to instantly analyze change between Spicules from two moments in time to see if malware is active (using the *Zero Form*). Subdivision of degree locations for the vectors around the equator leads mathematically to an almost infinite number of states that could be modeled. This is represented graphically in Figs. 3.1, 3.2, 3.3, and 3.4 below.

A *Zero Form* (Fig. 3.5), shown below as a round featureless ball, results when a Spicule at time T_1 is subtracted from a Spicule at time T_0 and no change has occurred in state variables being modeled by the tracking and fixed vectors. A *Zero Form* indicates that no malware is in operation.

Fig. 3.1 Equatorial view of Spicule, showing state variable vectors tracking normal or malware operation

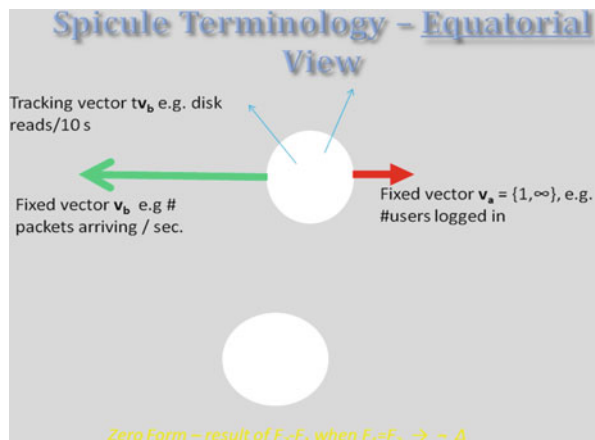


Fig. 3.2 Spicule showing port activity on a system
(*Normal Form*)

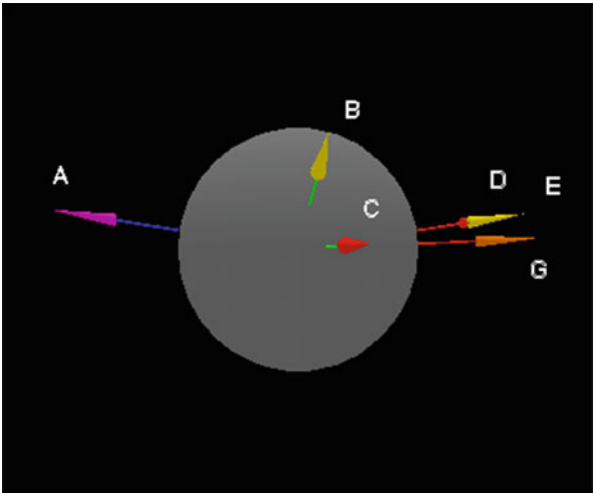
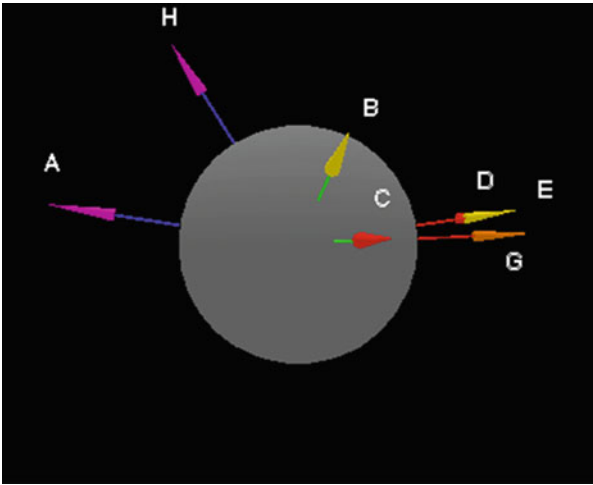
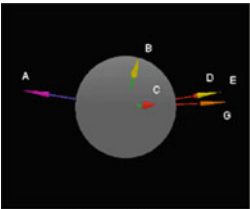
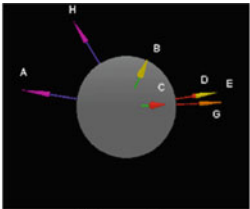
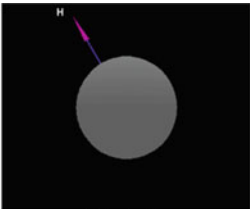


Fig. 3.3 Spicule showing a system under a SubSeven attack



 $-$  $=$ 

*Normal Form**Change Form**Attack Form*

Fig. 3.4 The mathematics of calculating the *Attack Form*

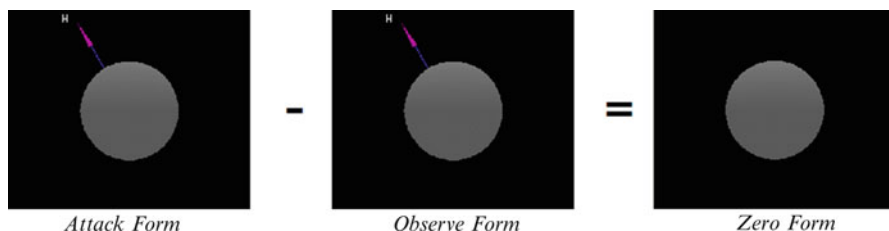


Fig. 3.5 Algebra for the identification of an attack

The Spicule approach is to display system activity or state variables in the form of vectors that project from the center of a sphere as in Fig. 3.2. 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%). Vectors can be mapped to represent any number of system state variables or log data that might be useful in detecting an attack.

3.3.5.1 Previous Work on Spicule Visualization Prototype

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 in specific, it does serve to illustrate the early concept and so is included. In testing, as any given port becomes opened, Spicule shows this by rendering a vector at the equator. As throughput increases on this port, the vector moves vertically up the sphere towards the northern pole. Figure 3.2 shows the purple-tipped vector pointing to the left; this vector is moving towards the northern pole as the activity on the SSH port (port 22) increases. Since it is just slightly above the equator, the activity is still relatively low which can be interpreted to mean that activity is characteristic of a system not under attack. In contrast, if the same vector were standing on the northern pole and pointing up, activity would be near maximum indicating a possibly dangerous system state. The Spicule in Fig. 3.2 is monitoring ports 22 (SSH, labeled A), 23 (Telnet, labeled B), 80 (HTTP, labeled C), 110 (POP3, labeled D), 137 NetBIOS (Name Service, labeled E), and 443 (HTTPS, labeled G). As the system's state changes, so will Spicule's. This generates a set of state variables.

To test this, a prototype used Backdoor SubSeven 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. Most commonly, attacks happen to ports 1243, 6776, and 27374. Figure 3.3 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 (labeled H) which has appeared suddenly with a great deal of traffic on an otherwise reserved port (1243).

In Figure 3.4 above, a *Normal Form*, *Change Form*, and *Attack Form* are illustrated. Finally, the mathematics of calculating the *Attack Form* and the relative reduction of data and interpretation of change is discussed below.

3.3.5.2 Mathematical Properties and Visual Algebra

The Spicule model is comprised of six unique states: *Normal Form*, *Zero Form*, *Change Form*, *Attack Form*, *Observe Form*, and *Predict Form*. These forms are generated utilizing the previously discussed section on vector mathematics. The *Normal Form* is the state in which the system is operating normally and not under attack. Opposite to this is the *Change Form*, which is a representation of a system under attack. The *Attack Form* is a signature (or isolated) view of an attack in progress that is occurring inside the *Change Form*. *Attack Forms* could be stored in a database for later reference, in which case they become *Predict Forms*, which are predictions of future attacks. The *Observe Form* is a state in which may or may not be an attack signature. Through mathematical operations, an *Observe Form* can be compared to a *Predict Form*. Each one of these forms has a unique visual appearance and mathematical signature. The algebra for each of these forms in Fig. 3.4 is listed in Table 3.2 and discussed in more detail below.

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, to isolate an attack and produce an *Attack Form*, simply subtract the *Normal Form* from the *Change Form* in Formula (1) above where *S* is the Spicule. The algorithm for this above process is listed below. Note that FOR EACH Vector (*i*) on the Spicule:

V_{Attack Form(i)} = V_{Normal Form(i)} - V_{Change Form(i)}

The visual representation of this algebra is presented in Fig. 3.4 above. Here, one can see the essence of the attack’s visual characteristics in the *Attack Form*. This is vector H. Such forms can be potentially stored into a database as the *Attack Form*

Table 3.2 Mathematical operations per Spicule model

Formula	Spicule model	Mathematical operation
(1)	Attack Form	S_{Attack Form}=S_{Normal Form} - S_{Change Form}
(2)	Observe Form	S_{Observe Form}=S_{Normal Form} - S_{Change Form}
(3)	Zero Form	S_{Zero Form}=S_{Attack Form} - S_{Observe Form}
(4)	Predict Form	S_{Predict Form} = S_{Normal Form} + S_{Attack Form}

of SubSeven or the family of malware that operates similar to SubSeven. Once they become stored and classified, they become our *Predict Form*.

An *Attack Form* is created from pre-classification of attack families for the major families of malware. They can be stored and used for identification. They would be one phase of this research. They are subtracted with a *Change Form* to classify an attack and thus respond 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 over time. This may also be stored in a database library of attacks for future use.

An *Observe Form* may or may not be 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 an *Attack Form* stored in a database to classify the family of attack that is occurring on the system. It is created from pre-classification of attack families for the major families of malware. These are stored and used for identification. They are subtracted with a *Change Form* to classify an attack and thus respond if a *Zero Form* results from the algebra.

A *Change Form* is always Spicule at time T_1 that a *Normal Form* (at time T_0) is subtracted from to calculate the *Observe Form*. One can detect an attack by using Formula (2) where S is the Spicule.

The major difference between these two formulas is that the latter (2) is used to create an *Observe Form*, which is a *possible Attack Form*, whereas the former (1) is used when creating an *Attack Form* only. The reasoning behind this is that Formula (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. Figure 3.6 shows the actual Spicules applied to Formula (2).

The *Observe Form* is potentially what an attack would look like while underway. It is compared against the *Attack Forms* to identify an attack. The method of performing this comparison is an algebraic subtraction as shown in Formula (3).

Here, S is the Spicule. Figure 3.6 below shows the actual Spicules applied to Formula (3). Note that this can easily be automated.

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 one method of how to watch for such an event if it occurs. The *Predict Form* is created by the additive property of the algebra. It is calculated using Formula (4): *Predict Form* = *Normal Form* + *Attack Form*.

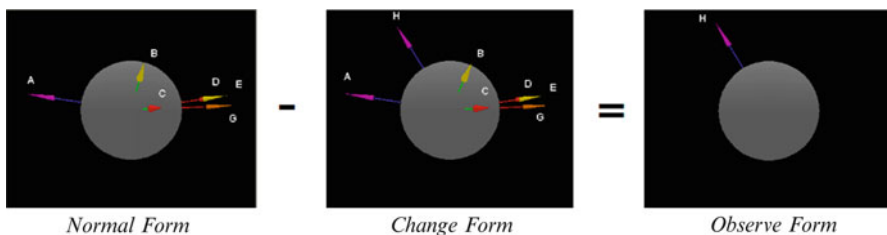


Fig. 3.6 *Observe Form* algebra

This produces what we expect the attack to look like if it occurs. The subtraction operation then identifies and confirms that the malware exists via:

$$S_{Zero\ Form} = S_{Predict\ Form} - S_{Change\ Form}$$

If a *Zero Form* exists, then the attack has been identified, classified, and can be responded to. It is important to note that a *Zero Form* occurs with subtraction of one set of state variable vectors from another when they exactly match or jitter control has been applied.

In the above example, the mathematics of Spicule produces a featureless Spicule (*Zero Form*) if the *Observe Form* equals the *Predict Form*. This drastically simplifies and speeds the process of recognition. The potential gain in identification time has the possibility to extend Spicule methodology to real-time visual and/or automated detection. This illustrates the impact of the analytic visual algebra because a security officer can look for the *Zero Form*, which dramatically displays that the system might be under attack.

3.3.6 *False Positive, False Negative Mitigation, and Jitter Control in FAST-VM Model*

The goal of this part of the FAST-VM will be to minimize the false report rate of the individual vector activity in the model. Vector location during normal operation and over time for a state variable will fluctuate. The FAST-VM algebra discussed previously would detect this as potential APT presence. Because these fluctuations—referred to as jitter—change over time, an approach to jitter control needs to be adaptive by the system. This method must operate such that normal jitter is differentiated from abnormal (APT) jitter and not flagged as a threat. There are several methods that might be implemented for mitigation: (1) sweep region adaption and (2) Bayesian probabilistic methods.

Sweep region adaption argues that tracking vectors—the ones that range from 0 to 100%—will have a region that they characteristically like to settle into based on time of day. For instance, the “CPU usage” state variable for a given host may range from 40 to 60% over a 12-hour period. This is referred to as its characteristic sweep region. Additionally, statistical methods can be employed on the vector to determine where it characteristically tends to be found, for example, 51% with a standard deviation of $\pm 2\%$. When conducting the FAST-VM Change Form analysis to detect state changes possibly due to APT, a vector for this state variable if falling within its typical sweep region would not be presented in the change form as an indicator of APT presence. The adaptive part to mitigate FPR and FRR is that if the sweep region is causing the vector to flag non-APT presence, or not to flag APT presence, the sweep region can be adjusted automatically as a variable in the algorithm generating the change form analysis.

Bayesian probabilistic methods can be utilized in a fashion akin to sweep region adaptation to predict and fine tune the probabilistic location of a state variable vector

based on where it was identified in the past. This method can also be made adaptive, and further statistical analysis can be performed not unlike the sweep region method.

3.3.6.1 Finite Angular State Transition-Velocity Machine

The Finite Angular State Transition-Velocity Machine (FAST-VM) extends the Spicule concept into a state model. Unlike other state models, this state model also models the velocity of change in the system state over time to create very advanced capabilities to capture the complex state changes created as malware operates in a system. Current methods cannot capably model the high order of state complexity and change that FAST-VM handles very easily. Additionally, the FAST-VM is one of the first methods to integrate all three major methods of performing intrusion detection (anomaly detection, misuse detection, and specification detection) into a unified model. This unification develops powerful synergies for malware classification and identification that have not previously existed.

3.3.6.2 Fast-VM Operation

The FAST-VM consists of Spicules as they transition over time (the anomaly detection at a given moment in time for high order state variable vectors) combined into N -dimension state transitions. Each transition has a velocity. The velocity is the rate of change in Spicule *Change Forms over time* and an attribute of probabilistic confidence that denotes the transitioned to state as a recognized state (such as one would find when APT modifies the state of the system). In each state of the graph, the Spicule algebra can be applied for analysis. The model looks as shown in Fig. 3.7.

In this example (Fig. 3.7), a variety of characteristics is evident:

1. Spicules representing state changes at various points in time,
2. A velocity equation $|h|$ (magnitude of the transition) that describes the transition speed from T_0 to T_1 ,
3. A cumulative *Attack Form* describing the attack signature for APT summed over time at T_3 ,
4. 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

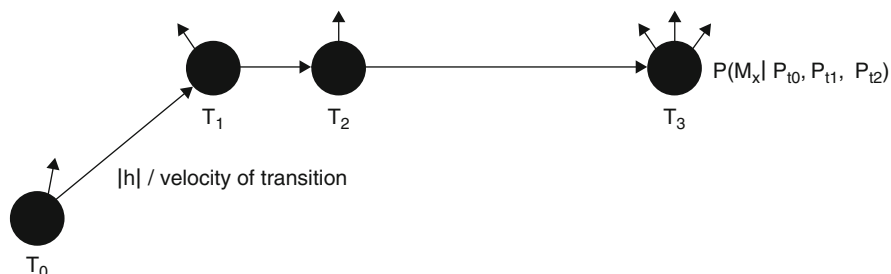


Fig. 3.7 A sample FAST-VM

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 identical.

The above diagram shows a FAST-VM creating a signature trail as APT is 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 as previously discussed. These attributes give FAST-VM a capacity to model and analyze very large amounts of data as it changes over time for the detection of the subtle changes in a system of hosts such as would be found with APT.

The above diagram also models a single category of APT as it is being analyzed. This becomes a known signature trail for this category. Notice the diagram is moving left to right in 2D space. FAST-VM is not limited to 20 signature trails. They could be created branching anywhere into 3D space creating an almost infinite modeling capacity. Unknown APT threat could be compared against signature branches shown above to rapidly classify the unknown APT into a category of previously seen APT without having to run the development of a full signature trail or branch. This offers the potential to stop an APT attack while it is instantiating and before it has taken hold.

The FAST-VM can do analysis at any given point in time using the Spicule mathematics and analyze *Spicule Attack Forms* in a time series sequence. Finally, it has a proposed method for dealing with jitter and classifications into families of malware using Bayesian probability, confidence, and velocity of transitions.

3.4 Application to Networks

FAST-VM is a powerful concept that can also be applied to entire networks of computers, not just a single host. There are two main strategies for applying FAST-VM to a network:

1. Consider each host's Spicule to be a vector on a larger "network spicule." This is useful for detecting attacks that affect many machines on a network at once.
2. Make a web of Spicules and analyze them all simultaneously. A standard network diagram can be adapted by replacing each system with a Spicule.
3. Deploy FAST-VM only on outward-facing gateways. This saves computational power because FAST-VM is not running on each host. It also reduces the workload for the admin because there are fewer Spicules to inspect. It would be effective at blocking external threats; however, it is not effective at detecting threats originating within the network.

Ultimately, it is up to the end user to decide where to deploy FAST-VM within their network, and there is no single correct way to do so. It will vary depending on the specific network. The basic FAST-VM algorithm stays the same when applied to networks. The only major difference between monitoring a single host and monitoring the network is the state variables and the methods of collecting them.

3.4.1 State Variables

FAST-VM relies on a list of state variables to create its Spicules and display information to the user. For FAST-VM to be useful, these state variables need to be well thought-out, specific attributes of a system that can be measured and analyzed. The following tables provide examples of the application of FAST-VM procedures in various industries and real-world scenarios.

In a practice application, variables are tailored based on specific function. Various systems can use the FAST-VM concepts including cars, medical devices, unmanned aircraft, and of course, personal computers and networks. In Table 3.3, the variables of intrusions detectable by FAST-VM are listed by device type. Table 3.4 lists FAST-VM intrusion detection for Automobile and Truck CAN ID systems and Table 3.5 defines intrusion detection capabilities for Network Protocols. Table 3.6 shows FAST-VM application in the health and medical industry listing Internet-Connected Medical Devices and types of intrusions detected. Table 3.7 defines

Table 3.3 Variables for individual computers or computer networks

Variable	Type of intrusion detected
Login frequency by day and time	Intruders may be likely to log in during off-hours
Frequency of login at different locations	Intruders may log in from a location that a specified user rarely or never uses
Time since last login	Break-in on “dead” account
Elapsed time per session	Significant deviations might indicate masquerader
Quantity of remote output	Excessive amounts of data transmitted to remote locations could signify leakage of sensitive data
Session resource utilization	Unusual processor or I/O levels could signal an intruder
Login failures	Attempted break-in by password guessing
Execution frequency	May detect intruders who are likely to use different commands, or a successful penetration by a legitimate user who has gained access to more privileged commands
Program resource utilization	An abnormal value might suggest injection of a virus or Trojan horse, which performs side-effects that increase I/O or processor utilization by a defined program.
Execution denials	May detect penetration attempt by individual user who seeks higher privileges
Read, write, create, delete frequency	Abnormalities for read and write access for individual users may signify masquerading or browsing
Records read, written	Abnormality could signify an attempt to obtain sensitive data by inference and aggregation
Failure count for read, write, create, delete	May detect users who persistently attempt to access sensitive data

Table 3.3 (continued)

CPU usage	DoS attack, malware activity
Open ports	Determine if a port being open is unusual
Metadata modification	Indicates an attacker is present on the system and could be injecting malware or doing other harmful work to the system.
Exhaustion of storage space	Denial of service attack; Malware might be present
Failure to receive SYN-ACK	The client’s machine is sending the SYN packet to establish the TCP connection and the web server receives it, but does not respond with the SYN/ACK packet. Can indicate a stealth scan
Half-open connections	Denial of service attack; can also indicate stealth scan

Table 3.4 Variables for automobile and truck CAN ID systems

Variable	Type of intrusion detected
Time interval between messages	Messages normally are generated at a specific interval. Any interval besides the set one is likely an attack
Volume of messages	Helps detect DoS attacks
Frequency of diagnostic messages	These are rare and generated by critical component failure. Frequent diagnostic message will rarely happen except as part of an attack
Car movement status (driving/idle)	This variable is combined with the one above indicate an attack, since diagnostic messages usually only appear while the car is idle

From “Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network” by Song et al. [25]

Table 3.5 Variables for network protocols

Variable	Type of intrusion detected
Number of illegal field values	Illegal values are sometimes user generated, but not in high quantities, assuming the user identifies and corrects their mistake
Number of illegal commands used	Illegal commands are sometimes user generated, but not in high quantities, assuming the user identifies and corrects their mistake
Field lengths	Helps detect buffer overflow vulnerabilities
Protocol or service not matching standard port/purpose	Occasionally a legitimate user will set up a service on a non-standard port, but it is far more likely that malware is attempting to use the port instead
Volume of data from destination to source	Useful in detecting DoS attacks
Network service used on destination	Some services will stand out as unusual

From “A hybrid approach for real-time network intrusion detection systems” by Lee et al. [26] and “intrusion detection tools and techniques: a survey” by Karthikeyan and Indra [27]

Table 3.6 Variables for internet-connected medical devices

Variable	Type of intrusion detected
Number of requests for patient controlled analgesic (PCA)	An acceptable range can be set for this value. Any deviation from this range indicates a problem
Defibrillator status (on/off)	Can be combined with other variables such as pulse rate or requests for PCA. (Note: Unconscious patients are unable to press the PCA button)
Pacemaker setting	Compared with pulse rate to determine if pacemaker is working properly
Pulse rate	Compared with pacemaker setting to determine if pacemaker is working properly
Blood pressure, oxygen saturation, respiration rate, and temperature	An acceptable range can be set for this value. Any deviation from this range indicates a problem
Standard deviation of vital signs sensors	Multiple sensors are often used to gather vitals. If one sensor is attacked to give a false reading, but not another, it will result in an increased standard deviation between the two

From “Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems” by Mitchell and Ing-Ray [28]

Table 3.7 Variables for armed unmanned aircraft systems (UAS)

Variable	Type of intrusion detected
Weapons ready (true/false)	Combined with location and status to be useful. For example, weapons should not be ready while taxiing
Location (target, airbase, non-target)	Combined with weapon status and flight destination
Thrust level	Combined with status. Each status should have a range of acceptable thrust levels so that fuel is not wasted
Status (taxi, transit, loiter, attack)	Combined with thrust. Each status should have a range of acceptable thrust levels so that fuel is not wasted
Landing gear status (up, down, error)	Combined with status and location; gear should not be up while taxiing or down while loitering over a target, for example
Flight destination (whitelisted, not whitelisted)	If the destination is set to a non-whitelisted location, it could be an operator error. Alternatively, it could be a third party trying to capture the UAV
Communication destination (whitelisted, not whitelisted)	If the comm’s destination is set to a non-whitelisted location, it could be an operator error. Or it could be a third party trying to intercept UAV communications
Standard deviation of redundant flight sensors	Multiple redundant sensors are often used to gather flight information (airspeed, altitude, etc.). If one sensor is attacked to give a false reading, but not another, it will result in an increased standard deviation between the two

From “Specification based intrusion detection for unmanned aircraft systems” by Mitchell and Ing-Ray [29]

Table 3.8 Variables for disk drives/storage devices

Variable	Type of intrusion detected
Modification of specific files	There are system executables, configuration files, log files, and system header files that shouldn't be modified, per the admin's definition
Modification of metadata, timestamps, or file permissions	Rarely done for legitimate purpose
Active disk time	Excessive active time could be a result of malicious activity
Numbers of hidden files or empty files	Rarely done for legitimate purpose, can be a sign of a race condition exploit in progress

From “Slick,” by Bacs et al. [30]

the application of FAST-VM with Unmanned Aircraft Systems (UAFs) as initially discussed in the introduction to this chapter. Finally, in Table 3.8, the use of FAST-VM for intrusion detection is presented for Disk Drives and Storage Devices.

3.5 Conclusion

In conclusion, computer security and the detection APTs is vital to strong e-commerce, military defense, aerospace, healthcare, financial institutions, and manufacturing industries [2–5]. In prior research, limitations of traditional Intrusion Detection Systems (IDS) were identified in the areas of human error, cost, and high error-rates due to large volumes of data being processed [2–9]. Current automated systems are restricted in the ability to recognize unusual states or attack states anomalies thus requiring a human analyst [4]. Yet humans have limited ability to consistently and effectively sift through large amounts of data which are the proficiency of computerized automated systems [4]. As cyber-crimes against business and society increase, automated systems to supplement human analysis are required to ensure safe secure networks and technologies [2, 3]. These findings and the information provided in this chapter support the need for alternative automated approaches to pattern recognition such as the FAST-VM for better analysis of real or perceived APT attack and present new technology [2, 3, 5–9].

The Finite Angular State Velocity Machine (FAST-VM) models and analyzes large amounts of state information over a temporal space. Prior development of the technology revealed capabilities of the FAST-VM to analyze 10,000,000 state variable vectors in around 24 ms. This demonstrates the application of “big data” to the area of cyber security. FAST-VM also unifies the three major areas of IDS (anomaly, misuse, and specification) into a single model. The FAST-VM mathematical analysis engine has shown great computational possibilities in

prediction, classification, and detection but it has never been instrumented to a system's state variables. In this chapter, the ability of the FAST-VM to map the state variables in a UAS system to detect APT as well as practical application in industry was examined.

References

1. Turner, J. (2016, September). *Seeing the unseen—Detecting the advanced persistent threat* [Webcast]. Dell SecureWorks Insights. Retrieved from <https://www.secureworks.com/resources/wc-detecting-the-advanced-persistent-threat>
2. Vert, G., Gonen, B., & Brown, J. (2014). A theoretical model for detection of advanced persistent threat in networks and systems using a finite angular state velocity machine (FAST-VM). *International Journal of Computer Science and Application*, 3(2), 63.
3. Dell SecureWorks. (2016, September). *Advanced persistent threats: Learn the ABCs of APTs – Part I*. Dell SecureWorks Insights. Retrieved from <https://www.secureworks.com/blog/advanced-persistent-threats-apt-a>
4. Daly, M. K. (2009, November). *Advanced persistent threat (or informational force operations)*. Usenix.
5. Ramsey, J. R. (2016). *Who advanced persistent threat actors are targeting* [Video]. Dell SecureWorks Insights. Retrieved from <https://www.secureworks.com/resources/vd-who-apt-actors-are-targeting>
6. Scarfone, K., & Mell, P. (2012). *Guide to intrusion detection and prevention systems (IDPS)* (pp. 800–894). Computer Security and Resource Center, National Institute of Standards and Technology.
7. Kareev, Y., Fiedler, K., & Avrahami, J. (2009). Base rates, contingencies, and prediction behavior. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 35(2), 371–380.
8. MacDonald, N. (2010, May). *The future of information security is context aware and adaptive*. Stamford, CT: Gartner Research.
9. Othman, Z. A., Baker, A. A., & Estubal, I. (2010, December). Improving signature detection classification model using features selection based on customized features. In *2010 10th international conference on intelligent systems design and applications (ISDA)*. doi: 10.1109/ISDA.2010.5687051
10. Eick, S., & Wills, G. (1993, October). Navigating large networks with hierarchies, In *Proceedings Visualization Conference '93* (pp. 204–210), San Jose, CA.
11. Han, G., & Kagawa, K. (2012). Towards a web-based program visualization system using Web3D. In *ITHET conference*.
12. Bricken, J., & Bricken, W. (1992, September). A boundary notation for visual mathematics. In *Proceedings of the 1992 IEEE workshop on Visual Languages* (pp. 267–269).
13. Damballa, Inc. (2010). *What's an advanced persistent threat?* [White Paper.] Damballa, Inc. Retrieved from https://www.damballa.com/downloads/r_pubs/advanced-persistent-threat.pdf
14. Erbacher, R., Walker, K., & Frincke, D. (2002, February). Intrusion and misuse detection in large-scale systems. In *IEEE computer graphics and applications*.
15. Vert, G., & Frincke, D. (1996). Towards a mathematical model for intrusions. In *NISS conference*.
16. Vert, G., Frincke, D. A., & McConnell, J. (1998). A visual mathematical model for intrusion detection. In *Proceedings of the 21st NISSC conference*, Crystal City, VA.
17. Vert, G., Chennamaneni, A., & Iyengar, S. S. (2012, July). A theoretical model for probability based detection and mitigation of malware using self organizing taxonomies, In *SAM 2012*, Las Vegas, NV.

18. Shuo, L., Zhao, J., & Wang, X. (2011, May). An adaptive invasion detection based on the variable fuzzy set. In *2011 international conference on network computing and information security (NCIS)*.
19. Hoque, M. S., Mukit, A., & Bikas, A. N. (2012). An implementation of intrusion detection system using genetic algorithm. *International Journal of Network Security & ITS Applications (IJNSA)*, 4(2), 109–120.
20. Vert, G., Gour, J., & Iyengar, S. S. (2010, November). Application of context to fast contextually based spatial authentication utilizing the spicule and spatial autocorrelation. In: *Air force global strike symposium cyber research workshop*, Shreveport, LA.
21. Chandran, S., Hrudya, P., & Poornachandran, P. (2015). An efficient classification model for detecting advanced persistent threat. In *2015 international conference on advances in computing, communications and informatics (ICACCI)* (p. 2003). doi:[10.1109/ICACCI.2015.7275911](https://doi.org/10.1109/ICACCI.2015.7275911)
22. Vert, G., & Triantaphyllou, E. (2009, July). Security level determination using branes for contextual based global processing: An architecture. In *SAM'09 The 2009 international conference on security and management*, Las Vegas, NV.
23. Vert, G., Harris, F., & Nasser, S. (2007). Modeling state changes in computer systems for security. *International Journal of Computer Science and Network Security*, 7(1), 267–274.
24. Vert, G., Harris, F., & Nasser, S. (2007). Spatial data authentication using mathematical visualization. *International Journal of Computer Science and Network Security*, 7(1), 267.
25. Song, H. M., Kim, H. R., & Kim, H. K. (2016). Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In *2016 international conference on information networking (ICOIN)*.
26. Lee, S. M., Kim, D. S., & Park, J. S. (2007). A hybrid approach for real-time network intrusion detection systems. In *2007 international conference on computational intelligence and security (CIS 2007)*.
27. Karthikeyan, K., & Indra, A. (2010). Intrusion detection tools and techniques—A survey. *International Journal of Computer Theory and Engineering*, 2(6), 901–906.
28. Mitchell, R., & Ing-Ray, C. (2015). Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems. *IEEE Transactions on Dependable and Secure Computing*, 12, 1.
29. Mitchell, R., & Ing-Ray, C. (2012). Specification based intrusion detection for unmanned aircraft systems. In *Proceedings of the first ACM MobiHoc workshop on airborne networks and communications—Airborne '12*.
30. Bacs, A., Giuffrida, C., Grill, B., & Bos, H. (2016). Slick. In *Proceedings of the 31st annual ACM symposium on applied computing – SAC '16. Computer Science and Network Security*, 7(1), 293–295. January 2007.

Gregory Vert is a US citizen, who specializes in advanced security research in the areas of authentication, malware detection, classification, and modeling of state changes caused by malware in a system. He is the inventor of the contextual security model and Spicule state change model for malware detection. He has extensive experience in industry as a software engineer and extensive security training from a variety of places such as Black Hat, DEFCON, SANS Hackers Exploits, and Wireless Security as well as having earned a CISSP security certification. He has held two security clearances, one during his military service and one while working for Boeing. He has taught soldiers from Fort Hood who attend Texas A&M and recently published a book defining the new field of Contextual Processing. As a part of his work he has developed a new model for security based on context referred to as Pretty Good Security that has the potential to be faster and more computationally efficient than existing methods. He is currently teaching cyber security at College of Security and Intelligence at Embry-Riddle Aeronautical University in Prescott, Arizona.

Ann Leslie Claesson-Vert is an Associate Clinical Professor in the School of Nursing, College of Health and Human Services at Northern Arizona University in Flagstaff, Arizona. Her expertise

lies in clinical research application, systems analysis, and development of innovative application of technology with a competency-based approach to practice in various industries. She also functions as an Assistant Professor at the Department of Medicine & Health Sciences George Washington University, Systems Analyst for the Higher Learning Commission, and Grant Peer Reviewers for the US Department of Health & Human Services, and Council for International Exchange of Scholars (CIES) – Fulbright Scholars program.

Jesse Roberts is an undergraduate student at Embry-Riddle Aeronautical University in the Cyber Intelligence and Security program. He assists Dr. Vert in researching and collecting state variables and developing the FAST-VM concept.

Erica Bott is an undergraduate student at Embry-Riddle Aeronautical in the Cyber Intelligence and Security program. She assists Dr. Vert in developing the FAST-VM concept and making it understandable for real people.