A Diagnosis Based Intrusion Detection Approach

Conner Jackson[1], Karl Levitt[1], Jeff Rowe[1], Srikanth Krishnamurthy[2], Trent Jaeger[3] and Ananthram Swami[4]
[1]Dept. of Computer Science, University of California Davis
[2]Dept. of Computer Science and Engineering, University of California Riverside
[3]Dept. of Computer Science and Engineering, Penn State University
[4]Army Research Laboratory, Adelphi, MD
cjackson@ucdavis.edu

***Abstract*—We describe preliminary work on a novel detection approach, which we call *diagnosis-enabled intrusion detection* (DEID), which takes a stream of evidence from multiple sources, aggregates the evidence and uses it to arrive at the "best" explanation for the observed activity. This approach requires the solution of four key scientific challenges: (i) a theory and algorithms for *monitor placement* that covers all system layers to prevent attackers from evading detection even when launching zero-day attacks; (ii) *evidence collection* for producing useful aggregated evidence from system actions in real-time without adversely affecting the mission; (iii) a theory of *diagnosis detection* for filtering and correlating evidence to test hypotheses regarding mission impact, producing both diagnoses and explanations of their causes; and (iv) *diagnosis presentation* for conveying explanations to domain experts to produce new knowledge to act on previously-unknown attacks effectively and to respond effectively to identified attacks that preserve mission requirements.**

I. Introduction

We develop a novel intrusion detection approach, diagnosis-enabled intrusion detection (DEID), which takes a stream of evidence from multiple sources, aggregates the evidence into a set of attack symptoms and uses those symptoms to arrive at the "best" explanation for the observed activity. We output a set of features that are understandable by a human analyst. Our approach subsumes signature-based detection; the evidence may map onto a known signature. It also subsumes anomaly-based and specification-based IDSs; it captures known anomalies and deviations from protocol semantics as symptoms. More importantly, our models capture correlated behaviors across levels (human actions, OS, applications and network) both for normal behaviors and under attack conditions. The fundamental question we wish to answer is *how best to combine observed behaviors, that have well-defined statistical properties, to develop a high quality intrusion detection system, which generates alerts with well-known semantic properties for both known and unknown attacks*?

Towards answering the above question, our diagnosis-enabling detection system will incorporate evidence from monitoring at multiple layers (human-centric or psychosocial, network protocols, OS kernel, and application) to serve as attack symptoms, principled statistical methods for aggregating symptom data streams into a tunable, high-quality diagnosis trigger, causation graphs to model how attacks and normal behavior are mapped onto symptom sets, and a principled method for explaining why the evidence is considered to be an attack on the mission.

Our approach is inspired by disease diagnosis in the medical community. Specifically, the goal in health monitoring is to quickly and accurately identify new *syndromes* (illness with no previously diagnosed cause) by distinguishing them from known diseases. Cases of a known disease are recognized by the presence of that disease's known symptoms and can be thought of as roughly equivalent to signature-based IDS. The main questions with syndromes for public health professionals are: (a) whether a previously unobserved set of symptoms represents a new disease, (b) whether this new disease is caused by variants of known diseases (e.g., mutating viruses), and (c) whether the cause of this disease is a major threat to the health of the population at large. To quickly identify new disease outbreaks, the public health system uses a technique called *syndrome surveillance*. Syndrome surveillance refers to methods to detect population (and individual) health changes that indicate the presence of previously undiagnosed illness.

The public health surveillance problem is nearly identical to the problem of mission-oriented intrusion detection incorporating risk. We develop our approach, inspired by syndrome surveillance, to solve problems that are not adequately addressed by the traditional IDS approaches, yet incorporates them as components. We use monitor evidence not as the triggers for an attack alert but only as a possible symptom of an attack. This symptom might also indicate some other rare but benign behavior that has not been previously observed. The problem is *we need sound and principled models for diagnosing the root cause of the observed set of symptoms to function as an explanation for previously unobserved activity*.

II. Related Work

Intrusion detection approaches can be predominantly classified into three classes: (a) signature-based detection, (b) anomaly detection, and (c) specification-based detection. With signature-based detection [17], [20], [1], [21], [13] attacks are inferred from specific *known* accompanying behavior. The obvious problem is that unknown, zero-day attacks, or even simple variants are undetected. Anomaly detection looks for sets of features that deviate from normally expected behaviors [25], [10], [8], [14], [3], [23]. The assumption that attacks are observable as statistical rarities leads to relatively high false positive rates due to rare yet benign use of the

system. Furthermore, alerts generated by anomaly detectors provide limited actionable intelligence for system administrators. Specification-based IDS systems assume *precise* protocol definitions [4], [11], [16], [19]) are available for identifying good behavior; everything else is an attack. The drawback is that specifications are not always available nor are they strictly adhered to by developers in practice. Several efforts employ AI or fuzzy logic to determine the likelihood of attacks (e.g., [15], [12]). Similar to anomaly-based approaches, these require training on both normal data and on attacks, and suffer from high false positives. Very few of these approaches take psychological or social level considerations into account in their detection models. Finally, work in intrusion correlation has primarily focused upon distributed alert correlation [5], [24] and typically depends upon specifics of the underlying alert mechanism [6], [7], [2], [18], [9], [22].

III. Problem Formulation

To address this problem, the first step is to formulate the relevant features of known attacks that have been previously observed that will be useful in diagnosing unknown attack activity. We begin with a set of previously known and implementable attacks, $A = \{a_1, a_2, \ldots\}$, letting a be an arbitrary attack. Define $AC = \{ac_1, ac_2, \ldots\}$ to be the set of attack classes, where ac refers to an arbitrary class of attack. The function ATC, defined as $ATC : A \rightarrow \mathcal{P}(AC)$, associates each attack with a set of attack classes. Given a system, S used to accomplish a certain mission, we wish to determine if S is operating normally with the mission requirements satisfied, is under attack from a known attack a, or is under an unknown attack of a particular class ac.

To formulate the behavior or the system, we define the set of known states system S can be in as $SS = \{sa_1, sa_2, \ldots sa_n, sn\}$, and ss which refers to an arbitrary state. The state sa_i, where $1 \leq i \leq n$, represents the state where S is under attack by corresponding attack a_i, and sn represents the state where S is operating normally. Let $CS(t)$ be an element of SS, $CS(t) \in SS$, that represents the current state of the system at time t. Each attack, a, is known and implementable, so we can experimentally observe S when $CS(t) = sa$. Since we define the meaning of the "normal operations" of S we can also experimentally observe S when $CS(t) = sn$. Thus, it should be noted that we can experimentally observe S in any state of SS, $CS(t) = ss$.

IV. Attack State Estimation using Dempster-Shafer Theory of Evidence

Our goal is to estimate the effect of the current attack state on the ongoing mission using a limited set of available symptoms as evidence. We wish to take into account both the presence and absence of evidence as well as lack of knowledge about potential evidence. To accomplish this, we apply the Dempster-Shafer Theory of Evidence to all potential observables to arrive at a belief in a certain attack state classification. The set of known states, SS, will play the role of the *frame of discernment* in Dempster-Shafer Theory (DST). In DST, it is assumed that the current state of S, $CS(t)$, is unknown to the observer, and must be one of the values from the frame of discernment. The value of $CS(t)$ is determined by providing, combining, and interpreting *beliefs* (sometimes called *evidence*) which take the form of a *basic belief assignment* (abbreviated *BBA*). All of these terms refer to the same object, namely a function that maps, over a particular frame of discernment, all possibles subsets of the frame of discernment to a *belief mass* or *mass* (a real value between 0 and 1, inclusive), defined as $BBA_{SS} : \mathcal{P}(SS) \rightarrow [0, 1]$. The total mass of a belief must add to 1, $\sum_{i \in \mathcal{P}(SS)} BBA_{SS}(i) = 1$, and the empty set must have no mass, $BBA_{SS}(\emptyset) = 0$. The purpose of beliefs is to reason about, and provide evidence for, the value of the $CS(t)$. The belief mass assigned to each subset of SS is interpreted as a subjective probability (or "opinion") that the given subset contains the $CS(t)$. Say you have a non-empty subset of the frame of discernment, $E|E \subseteq SS \wedge E \neq \emptyset$, a belief mass, X, and a belief function B_{SS} such that $B_{SS}(E) = X$. If $X = 0$, you don't believe that E contains the $CS(t)$, and if $X = 1$ you believe absolutely that E must contain the $CS(t)$. Values of X in-between the extremes, $0 < X < 1$, represent the varying strengths of opinion. Note when $X = 0$ you are not giving any opinion towards E *not* containing the $CS(t)$. To accomplish this effect, evidence should be assigned to the complement of E instead, $B_{SS}(SS - E) = X$. From the above explanation, one can see why $B_{SS}(\emptyset) = 0$, as we cannot provide evidence towards there being no $CS(t)$, since, by definition, there is always a current state. One can also see that assigning mass towards $B_{SS}(SS) = X$ tells us that "there exists a current state," which is effectively nothing when attempting to determine $CS(t)$. Mass assigned in this manner allows us to represent the lack of knowledge or strength in a belief.

V. Observables

To collect beliefs that will assist in determining the $CS(t)$, one will first need to gather a set of *observables*. Let $O = \{O_1, O_2, \ldots\}$ be the set of observables, where o is a particular one. Each observable is a function taking a time, t, and returning a real value, $o(t)$. Conceptually, an observable represents any process that can be monitored or sampled to produce a time series. For instance, the speed of a particular vehicle could be an observable, as its speed is defined and can be measured at each point in time. This definition is quite broad, so we will restrict our set, O, to only contain *relevant* observables. Relevant, in this context, means the values of o are not the same, or very similar to one another, when the CS varies across the values of SS. In other words, to be relevant, the value of o must be dependent on the value of the CS. For example, if we assume S is a networked computer, $SS = SF, N$, where SF is the state where S is under attack by a Synflood, and N is the normal state where S is idle. If o is the speed of a particular vehicle, we can see that whether $CS = SF$ or $CS = N$ the value of o will not be affected, and thus is *irrelevant*. However, if o was the network traffic coming into S, we can see that if $CS = SF$ would likely see larger values of o than $CS = I$, and is hence *relevant*.

Since all states in SS are replicable experimentally, it should be possible to determine which o are relevant.

VI. Symptoms

A *symptom* takes, as input, the value of an associated observable, and emits, as output, a belief over SS that provides an opinion of the CS. It is assumed the symptom is fed newly sampled values from the associated observable at regular, static, time intervals. The static time-interval requirement isn't necessary, but simplifies discussion. A symptom is allowed to maintain and make decisions based on an internal state, so the same observable value as input can yield different results based on the values input before it. We will assume we have a set of constructed $S = \{s_1, s_2, \dots\}$, where s is a particular symptom. A symptom's associated observable is determined by a user-defined mapping function SO, which takes a symptom s, and maps it to its observable from O, o. Note that multiple symptoms can be mapped to the same observable, and it is assumed each observable is mapped to by at least one symptom. The value of a symptom at time t is given by $s((SO(s))(t))$, assuming that s has been called for all previous values of t (necessary since, again, s maintains an internal state). For clarity, we will take $s(t)$ to represent the value of s at time t following the operation outlined above.

A symptom, as given above, can be thought of like a medical symptom. Suppose a doctor is attempting to diagnose a patient who feels sick. The patient, in this case, is the system S. The patient may have any number of illnesses, as well as be healthy, forming the set of known states, SS. The doctor does not know in advance what the patient's medical state is, CS. A possible observation, o, the doctor could make is the patient's level of nasal congestion. A symptom, s, associated with o, would be the presence of a "stuffy nose." In this case, if the nasal congestion was severe enough ($o(t)$ was above a threshold), then the doctor might have a high belief, B_{SS}, the patient could have the common cold or allergies, $B_{SS}(\{COMMON\_COLD, ALLERGIES\}) = 0.9$. If nasal congestion was not present, the doctor may have a low belief that the patient is healthy $B_{SS}(\{HEALTHY\}) = 0.05$, since the patient arrived under the premise of being sick, and a higher belief that the patient could be experiencing the stomach flu $B_{SS}(\{STOMACH\_FLU\}) = 0.2$. In both cases, the remainder of the doctor's belief mass would be allocated to the frame of discernment, which might represent his lack of experience in the area he is attempting to diagnose, his eagerness to go home, or his dislike of the patient. Each of these reduces the confidence in his results.

A class of symptom, we will refer to then as *binary* symptoms, adheres to the above specification, is simple to construct, and operates intuitively like the above example. Binary symptom operation begins with the current observable value, $o(t)$, which is fed into the symptom. This value is then fed into a decision algorithm, DA. DA takes a real input value and outputs a boolean, $DA : \mathbb{R} \to \{T, F\}$, and may choose to hold an internal state. If the return value of $DA(o(t))$ is true, then a precomputed belief, BT, is returned as the symptom's belief. Otherwise, a precomputed belief, BF, is returned instead. If DA evaluates to true the symptom is said to be *present*, otherwise the symptom is *absent*.

VII. Symptom Combination

We have the set of symptoms we have constructed, S, and we now wish to combine the individual beliefs produced by calling all $s(t)$ at time t. In DST, belief combination is accomplished through fusion operators. An arbitrary fusion operator, FO, takes two beliefs as input and outputs a single, fused belief as output. These operators can also be applied to a set of beliefs, $BS = \{bs_1, bs_2, \dots\}$, by starting with an empty belief, $B|B_{SS}(SS) = 1$, and applying the fusion operator to the last fused belief (or initial empty belief) and the next belief from BS, $r_1 = FO(B, bs_1), r_2 = FO(r_1, bs_2), \dots$, until all beliefs have been combined to form the final output belief, r_n.

Our situation is not as simple as selecting a single fusion operator to combine all of the beliefs generated from S. This is because some symptoms of S may or may not be dependent on one another, and fusion operators are built assuming sets of exclusively dependent or independent beliefs. To resolve this, we introduce the set of dependent symptom sets, $DSS = \{dss_1, dss_2, \dots\}$, where dss is an arbitrary dependent symptom set. Each symptom s in S should appear in exactly one dss and no other. One can imagine the symptoms contained in each dss being the evaluations of a single doctor or sensor. Each symptom in a dss should be combined using a fusion operator intended for dependent beliefs. Once all dss sets have been combined into beliefs, these resulting beliefs can then be combined using an fusion operator intended for independent data, and will result in a single final belief.

A tested pair of fusion operators that seem to produce good results are the averaging and cumulative fusion operators constructed by Audun Jøsang, Javier Diaz, and Maria Rifqi in the paper "Cumulative and Averaging Fusion of Beliefs." The averaging fusion operator, AVG, is intended to combine dependent beliefs, A and B, and is given by

$$AVG_{SS}(x) = \frac{A_{SS}(x)B_{SS}(SS) + B_{SS}(x)A_{SS}(SS)}{A_{SS}(SS) + B_{SS}(SS)}$$

$$AVG_{SS}(SS) = \frac{2A_{SS}(SS)B_{SS}(SS)}{A_{SS}(SS) + B_{SS}(SS)}$$

Assuming that both $A_{SS}(SS) \neq 0$ and $B_{SS}(SS) \neq 0$, otherwise a separate set of rules must be applied. The cumulative fusion operator, CUM, is intended to combine independent beliefs, and is given by

$$CUM_{SS}(x) = \frac{A_{SS}(x)B_{SS}(SS) + B_{SS}(x)A_{SS}(SS)}{A_{SS}(SS) + B_{SS}(SS) - A_{SS}(SS)B_{SS}(SS)}$$

$$CUM_{SS}(SS) = \frac{A_{SS}(SS)B_{SS}(SS)}{A_{SS}(SS) + B_{SS}(SS) - A_{SS}(SS)B_{SS}(SS)}$$

and holds the same constraints as the averaging fusion operator. The equations where the frame of discernment is allocated zero belief mass are omitted, as the binary symptoms described above will never experience this case. This was done by design, as it always allows later beliefs to provide evidence contrary to current beliefs. If a belief ever has zero mass assigned to it, its assumed that all non-zero elements of

belief *must* contain the CS, and all new evidence contrary to these elements is ignored. Enforcing a non-zero frame of discernment follows the philosophy, as Pliny states, "the only certainty is that nothing is certain," a sound view when attempting to detect the unknown. Of course, if S can *provably* be constricted to the non-zero elements of a given belief, then assigning zero to the frame of discernment is the best course of action, but knowledge of this strength is beyond what can be provided by the described system.

VIII. Interpretation

After fusion, we are left with a single combined belief that should reflect the views emitted by our symptom set. This belief can be used to estimate the subjective probabilities of any event over the associated frame of discernment. This requires the Dempster-Shafer belief, Bel, and plausibility, Pl, operations, which form the lower and upper bounds on the probability of an event. The belief operation, Bel, takes a belief, b, and an *event*, e, (a subset of the beliefs associated frame of discernment) and sums together the masses of all other events, t, that are a proper subset of e, $Bel(b,e) = \sum_{t \subseteq e} b(t)$. The belief operation adds together the evidence supporting that e contains the CS to form a lower bound on its probability. The plausibility operation, Pl, also takes a belief and event. However, it instead sums together the beliefs masses of all events that intersect with elements of e, $Pl(b,e) = \sum_{t \cap e \neq \emptyset} b(t)$. The plausibility operation adds together all evidence that doesn't contradict that e is the CS, and forms an upper bound on its probability. From these operations, we can see the subjective probability of e, $P(e)$, is bounded by $0 \leq Bel(b,e) \leq P(e) \leq Pl(b,e) \leq 1$.

We now have the tools necessary to determine if S is operating normally, under attack from a known attack, or under attack from an unknown of a particular class. Remember, an event represents a set of states, and the subjective probability computed from that state shows our confidence that this set contains the CS. Thus, to estimate the probability that S is currently operating normally, we compute for $e = \{sn\}$. To estimate the probability that S is under attack from one of our known attacks, sa, we compute for $e = \{sa\}$. Finally, to estimate that S is under an unknown attack of a known class, ac, we compute for $e = \{\forall s \mid ac \in ATC(s)\}$. To determine which of the events computed is above is most likely, the ranges produced by each, $[Bel(b,e), Pl(b,e)]$, should be compared. A possible comparison function could take the average of the lower and upper bounds, and take the largest among them. Another could compare the upper bound, and if both were equal, compare the lower bound to determine the largest. Yet another could take into account the difference between the upper and lower bounds, the uncertainly in our probability estimate, to select more or less precise solutions (this value could also be used to determine if we needed to gather more evidence to shrink this bound). Whatever this comparison function emits is taken to be the predicted "state" (not equivalent to the possible states contained in SS, since we have also included attack classes) of S at time t.

IX. Experimental Validation

To validate our DEID approach, we perform experiments on the DETER testbed. This testbed is specifically designed for cyber-security experimentation and consists of hundreds of computers made available remotely using a web-based management interface. This allows us to investigate cross-layer symptoms from attacks implemented at a single layer. These dependencies are typically not available with high-fidelity in simulation.

We design a simple network configuration composed of two LANs: one LAN with attacking hosts and one with the victim as well as observer nodes for monitoring. We instrument testbed nodes with monitors at the operating system level building on standard OS accounting tools for system performance and fault monitoring. These tools regularly poll kernel data structures and extract important values as observable OS-level symptoms. We collect symptoms at the application level using standard application logging functionality. In addition, we employ network monitors using an observer node and packet monitor at the LAN gateway router.

For our experiment, we investigate a large number of cross-layer symptoms produced by known attacks. For this initial validation, we use three different TCP denial-of-service techniques, each implemented on a single attack node. First is the traditional Synflood which connects to the victims multiple times without completing the three-way TCP handshake. This causes the victim's TCP connection table in the kernel to be consumed with partial sessions that never terminate, preventing legitimate connections from completing due to lack of space. The second attack is TCP SockStress which is a testing tool for determining a systems ability to tolerate a variety of malformed TCP conditions. Thirdly, we use the SlowLoris tool which fills up all available HTTP connections with requests that have very slow response times. This is a stealthy attack with no record produced in application logs.

The hypothesis we wish to test with our experiment is whether the previous three known DoS attacks manifest symptoms that are useful in detecting an unknown attack whose observable behavior fails to match the known attack behavior. To accomplish this, we design a new TCP DoS attack using a combination of techniques from the known attacks that are performed at a rate far below that which is necessary for denial-of-service individually. Launching these stealthy versions of known attacks simultaneously causes denial-of-service without triggering detectors configured to detect the known version. Our hypothesis is that this new attack will share enough of the same symptoms with a union of the three known attacks to provide a diagnosis even though the known signatures fail.

X. Preliminary Results

We run multiple instances of the three known attacks: TCP Synflood, SockStress, and SlowLoris, observing and recording the presence or absence of approximately 5000 potential symptoms we identify at four system layers. We collect OS level accounting statistics, TCP flag statistics at the router, connection response behavior at the network LAN observer,

and web application log behavior. Out of this large number of candidate symptoms, we identify 32 that show differential response to the known attacks; that is, only 32 symptoms are enough to allow for disambiguation of the known attack types. In addition, we emulate normal traffic to our web server application using a variety of common web benchmarking and testing tools that make patterns of requests to locations in the document tree at various rates. Finally, we launch the unknown attack and observe the set of symptoms that are produced Figure 1 shows symptoms and their occurrence in the presence of the five experimental system states. We show only the 14 most important of the 32 symptoms used in the analysis for clarity and brevity.

Sensor	Symptom	Normal	Synflood	Sockstress	Slowloris	Unkown
Apache Status Log	Closing Connection	~	X	O	~	O
Apache Status Log	Logging	O	O	X	X	X
Apache Status Log	Waiting For Connection	O	X	X	X	X
Gateway	IPv4 Checksum	O	O	X	X	X
Gateway	IPv4 Flag DF	O	X	O	O	X
Gateway	TCP Flag ACK (Low)	O	X	X	X	X
Gateway	TCP Flag ACK (High)	O	O	O	X	O
Gateway	TCP Window Size (Low)	O	X	O	X	O
Gateway	TCP Window Size (High)	O	O	X	X	X
Observer	Response Time	O	O	X	X	X
Observer	Return Code	O	O	X	X	X
Server Metric Sampler	LOADAVG_ActiveProc	~	O	X	X	X
Server Metric Sampler	NETSTAT_TcpExt_ListenDrops (Low)	~	O	~	X	X
Server Metric Sampler	NETSTAT_TcpExt_ListenDrops (High)	O	X	~	O	X

Fig. 1. Table of the symptoms for 5 experimental system states. The colors of the rows indicate the layer where the symptoms were collected: (red: application, blue: router, green: observer node, purple: OS)

A. DST State Assignment for Known Attacks

For each symptom, we assign a belief mass for its occurrence in conjunction with the known attacks. We also assign a belief mass for its presence in the case of normal traffic. Finally, to asses whether a particular symptom might indicate a denial-of-service class of attack against a mission critical service, we define a belief that it is present in a generic denial-of-service attack using its rate of occurrence in the known attack types. Then, for each experimentally collected symptom set, we compute the belief for each of the five system states using the Dempster-Shafer rule of combination over time as evidence is collected. We judge the effectiveness of our DST algorithm in assigning belief in the correct state. In figure **??** we show the results for the DST plausibility over time as the SlowLoris stealthy DoS attack is launched. Each graph shows the plausibility for a different system state hypothesis using the same set of observable symptoms as input. We execute the SlowLoris attack on its slowest, most stealthy setting in this example to observe the ability of DST to detect and distinguish SlowLoris from the other known attacks and from normal traffic. One notices that it takes more than 50 seconds before this attack method is effective. During that time, the hypothesis of the SlowLoris state proceeds from very low (plausibility = 0.4) to high (plausibility $>$ 0.9 at $t = 75$ seconds). While initially the Synflood hypothesis is higher than SlowLoris (plausibility = 0.6), as the SlowLoris attack proceeds, the plausibility of the Synflood state drops as more defining symptoms are received. Also, the plausibility of the normal traffic state hypothesis starts at near certainty (plausibility $>$ 0.98), then drops and remains low just as the SlowLoris state hypothesis becomes most plausible. These results show that the DST belief assignment of system state works well in distinguishing among known attacks and normal traffic based upon the same set of symptoms. Similar results were obtained during the execution of the other known attacks with a variety of rates and during times of normal traffic behavior. Our DST approach correctly assigned belief values in all cases.

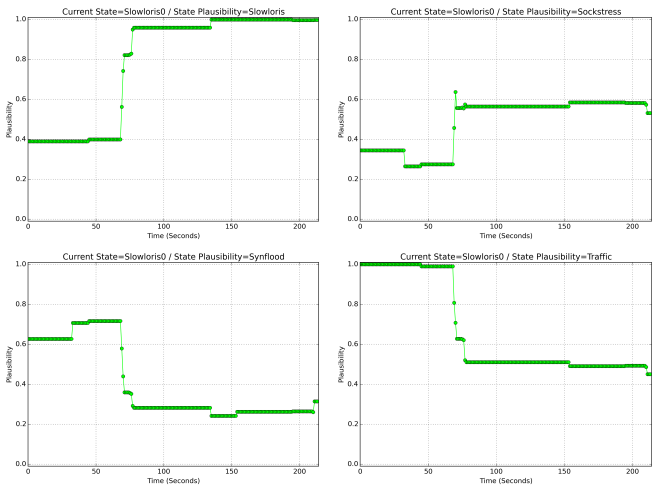

Fig. 2. Graphs showing the DST plausibility over time during the course of a SlowLoris attack.

B. Attack Classification of Unknown Attacks Using DST

The critical question remaining is how, using the symptoms identified from known attacks, does our approach respond in the presence of an unknown attack? To investigate this, we launch our custom stealthy TCP denial-of-service attack which uses a combination of techniques from the known attacks but at a much lower rate, which fail to trigger any of the available signature-based detection methods. Then we track the belief that the system state is consistent with an unknown TCP denial-of-service attack using the entire symptom set. The top two graphs of figure 3 show the DST plausibility for the system state being subject to an unknown TCP DoS attack and the plausibility of normal traffic. Almost immediately, enough symptoms are present to notice the unknown attack state. In this case, however, the plausibility of normal traffic remains high until several seconds into the attack when a slowdown in the targeted service begins to be felt at around 30 seconds. In practice, a security administrator or automated cyber-maneuver procedure would be able to take action based upon the change in unknown attack belief regardless of the belief in normal traffic if the targeted resource were mission critical. For comparison we also show the plausibility for two of the known attacks which remain low as desired.

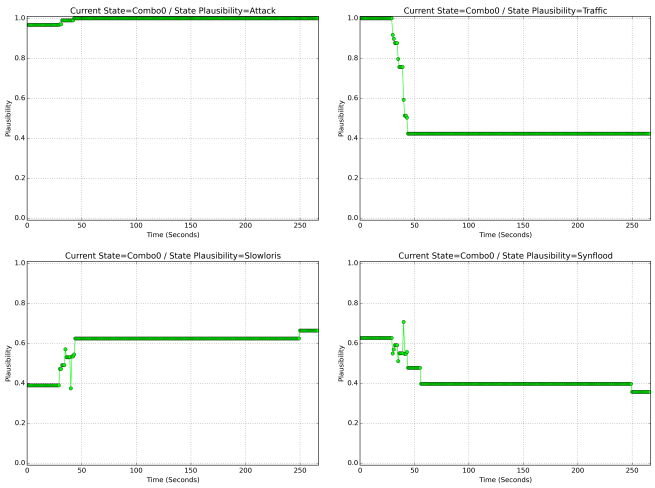

Fig. 3. Graphs showing the DST plausibility over time during the course of a new, unknown attack.

XI. Conclusion

We have developed a novel diagnosis-based intrusion detection procedure that applies the Dempster-Shafer theory of evidence to observable symptoms to detect unknown attacks and provide a useful classification. We show a proof-of-concept implementation and it's performance with TCP denial-of-service attacks. We wish to emphasize that our goal is not to develop an new DoS attack detector but rather to use DoS as an initial test case for validation of our approach.

Our ongoing work applies this approach to other attack types which are less evident than DoS, such as SQL injection and web browser hijacking. In addition, we investigate methods for automatic generation of symptom sets using high level security specifications and static analysis of code to identify critical data structures in the kernel, and in applications that provide non-bypassable evidence that aids in proper attack classification.

XII. Acknowledgments

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] Hyang ah Kim. Autograph: Toward automated, distributed worm signature detection. In *In Proceedings of the 13th Usenix Security Symposium*, pages 271–286, 2004.

[2] F. Anjum, D. Subhadrabandhu, and S. Sarkar. Intrusion detection for wireless adhoc networks. In *Proceedings of Vehicualar Technology Conference, Wireless Security Symposium*, 2003.

[3] Debojit Boro, Bernard Nongpoh, and Dhruba K. Bhattacharyya. Anomaly based intrusion detection using meta ensemble classifier. In *Proceedings of the Fifth International Conference on Security of Information and Networks*, SIN '12, pages 143–147, New York, NY, USA, 2012. ACM.

[4] C.Y.Tseng, P.Balasubramanyam, C.Ko, R.Limprasittiporn, J.Rowe, and K.Levitt. A specification-based intrusion detection system for AODV. *ACM Workshop on Security in Ad hoc and Sensor Networks (SASN)*, 2003.

[5] D.Sterne, P.Balasubramanyam, D.Carman, B.Wilson, R.Talpade, C.Ko, R.Balupari, C.Tseng, T.Bowen, K.Levitt, and J.Rowe. A general cooperative intrusion detection architecture for MANETs. In *3rd IEEE International Workshop on Information Assurance*, 2005.

[6] D.Subhadrabandhu, S.Sarkar, and F.Anjum. Efficacy of Misuse Detection in Ad hoc Networks. In *IEEE SECON*, 2004.

[7] D.Subhadrabandhu, S.Sarkar, and F.Anjum. RIDA: Robust Intrusion Detection in Ad hoc Networks. In *IFIP Networking*, 2005.

[8] Wenliang Du and Lei Fang. Lad: Localization anomaly detection for wireless sensor networks. In *In Proceedings of the 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS '05*, pages 874–886, 2005.

[9] F.Anjum and R.Talpade. Packet-Drop Detection Algorithm for Ad hoc Networks. *IEEE VTC*, 2004.

[10] Alexandros G. Fragkiadakis, Vasilios A. Siris, and Nikolaos Petroulakis. Anomaly-based intrusion detection algorithms for wireless networks. In *Proceedings of the 8th international conference on Wired/Wireless Internet Communications*, WWIC'10, pages 192–203, Berlin, Heidelberg, 2010. Springer-Verlag.

[11] Mathew Graves and Mohammad Zulkernine. Bridging the gap: software specification meets intrusion detector. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, PST '06, pages 31:1–31:8, New York, NY, USA, 2006. ACM.

[12] A. Grediga, F. Ibarra, F. Garcia, B.Ledesma, and F. Brotons. Application of neural networks in network control and information security. In *LNCS*, 2006.

[13] Fabian Hugelshofer, Paul Smith, David Hutchison, and Nicholas J. P. Race. OpenLIDS: a lightweight intrusion detection system for wireless mesh networks. In *MOBICOM*, pages 309–320, 2009.

[14] Edward Kaiser, Wu-chang Feng, and Travis Schluessler. Fides: remote anomaly-based cheat detection using client emulation. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 269–279, New York, NY, USA, 2009. ACM.

[15] C. Katar. Combining multiple techniques for intrusion detection. In *International Journal of Computer Science and Network Security*, 2006.

[16] Konstantinos Kemalis and Theodores Tzouramanis. SQL-IDS: a specification-based approach for SQL-injection detection. In *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, pages 2153–2158, New York, NY, USA, 2008. ACM.

[17] Christopher Kruegel and Thomas Toth. Using decision trees to improve signature-based intrusion detection. In *Proceedings of the 6th International Workshop on the Recent Advances in Intrusion Detection (RAID 2003)*, pages 173–191. Springer Verlag, 2003.

[18] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. *ACM MobiCom*, Aug. 2000.

[19] Robert Mitchell and Ing-Ray Chen. Specification based intrusion detection for unmanned aircraft systems. In *Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications*, Airborne '12, pages 31–36, New York, NY, USA, 2012. ACM.

[20] James Newsome. Polygraph: Automatically generating signatures for polymorphic worms. In *In Proceedings of the IEEE Symposium on Security and Privacy*, pages 226–241, 2005.

[21] Vern Paxson. Bro: A system for detecting network intruders in real-time. In *Computer Networks*, pages 2435–2463, 1999.

[22] R.Rao and G.Kesidis. Detecting Malicious Packet Dropping Using Statistically Regular Traffic Patterns in Multihop Wireless Networks that are not Bandwidth Limited. *IEEE Globecom*, 2003.

[23] Mahbod Tavallaee, Natalia Stakhanova, and Ali Akbar Ghorbani. Toward credible evaluation of anomaly-based intrusion-detection methods. *Trans. Sys. Man Cyber Part C*, 40(5):516–524, September 2010.

[24] Y.Huang and W.Lee. A Cooperative Intrusion Detection System for Ad hoc Networks . In *ACM Workshop on Security of Ad hoc and Sensor Networks (SASN)*, 2003.

[25] Yongguang Zhang, Wenke Lee, and Yi-An Huang. Intrusion detection techniques for mobile wireless networks. *Wirel. Netw.*, 9(5):545–556, September 2003.