Cross-Level Behavioral Analysis for Robust Early Intrusion Detection

Shun-Wen Hsiao, Yeali S. Sun
Dept. of Information Management
National Taiwan University
Taipei, Taiwan
{r93011, sunny}@im.ntu.edu.tw

Meng Chang Chen
Institution of Information Science
Academia Sinica
Taipei, Taiwan
mcc@iis.sinica.edu.tw

Hui Zhang
School of Computer Science
Carnegie Mellon University
Pittsburgh, USA
hzhang@cs.cmu.edu

***Abstract*—We anticipate future attacks would evolve to become more sophisticated to outwit existing intrusion detection techniques. Existing anomaly analysis techniques and signature-based detection practices can no longer effective. We believe intrusion detection systems (IDSs) of the future will need to be capable to detect or infer attacks based on more valuable information from the network-related properties and characteristics. We observed that even though the signatures or traffic patterns of future stealthy attacks can be modified to outwit current IDSs, certain behavioral aspects of an attack are invariant. We propose a novel approach that jointly monitors network activities at three different levels: transport layer protocols, (vulnerable) network services, and invariant anomaly behaviors (called *attack symptoms*). Our system, SecMon, captures the network behaviors by simultaneously performing cross-level state correlation for effective detection of anomaly behaviors. For the most part, the invariant anomaly behavior has not been fully exploited in the past. A probabilistic attack inference model is also proposed for attack assessment by correlating the observed attack symptoms to achieve the low false alarm rate. The evaluations demonstrate our prototype system is efficient and effective for sophisticated attacks, including polymorphism, stealthy, and unknown attack.**

Anomaly Detection, Attack Assessment, Cross-Level Behaviorial Analysis, Network Protocol, Finite State Machine

I. INTRODUCTION

Internet attacks that exploit the vulnerabilities of software, computers and networks account for a substantial portion of the security incidents. It can damage the computer systems, disrupt the network services or block network traffic [1], [2]. Worst, many attacks are capable of installing malicious programs on infected hosts for later invasion [3]–[6]. Different from computer virus, these attacks require no human involvement and can compromise and propagate by themselves [2].

Many existing anomaly detection practices are based on the outbreak phenomena of attacks, e.g., by observing abnormal traffic volumes or the dispersion of IP addresses [7], [8]. They first establish the normal profile of the network, and check if the current network traffic patterns violate the normal profile. Although it is possible to detect unknown attacks with similar spreading pattern, the severe damages have already made. Further, it may have high false positive rate because being different from normal profile does not necessarily imply an attack.

This work was partially supported by the iCAST project sponsored by the National Science Council, Taiwan, under the Grant No. NSC97-2745-P-001-001.

Some other research works adopt the Susceptible-Infectious (SI) Epidemic model to formulate the global spreading of an Internet worm [1], [9], [10]. Among them, in [9], the authors proposed a discrete time model for early detection of global worm propagation. Although it can detect worms when only a small portion of population in the Internet is infected, it is not that suitable for small networks. Moreover, before global trend presents, lots of hosts may have been compromised.

While these approaches may have been effective before, we anticipate the future attacks may evolve to become more sophisticated. For example, they may adopt stealthy strategies without being detected. A stealthy attack may perform selective or intelligent probes to find possible victim hosts, which is different from traditional exhaustive try-and-error strategies. A stealthy attacker can even set up a hostile server waiting for vulnerable hosts to contact it [2]. Also, sophisticated attackers have started to make polymorphic attacks which mutate the appearance of packet in every replicated instance to avoid being detected by signature-based IDSs [11].

Software vulnerabilities are basically came from design errors and programming flaws. Although various bug-finding and design verification tools have been developed, they are basically static analysis and are difficult in tackling increasing attack threads in runtime intrusion detection [12]. The number of newly discovered vulnerabilities reported to CERT/CC continues to grow more than double each year [13]. Moreover, the presence of new and stealthy attacks and the emerging of polymorphic attacks impose unprecedented threats and challenges to network security [1], [14].

Besides the characteristics such as invariant bytes and rapid scanning of attacks which are commonly used in IDSs, we observe a number of network attack such as Blaster [3], Sasser [4] and Welchia [15], possess sophisticated procedure-based behavior. The attacker undergoes a sequence of network interactions to compromise the victim. They may include the execution of communication protocol, accessing network service, and possibly installing malicious program. We notice that there are some behaviors that will underline the characteristics of an attack. They are referred to as the *attack symptoms* in this paper.

Fig. 1 depicts the attack procedure of Blaster. In this example, we observe that the interaction between the attacker and the victim involves activities at three different levels. At the

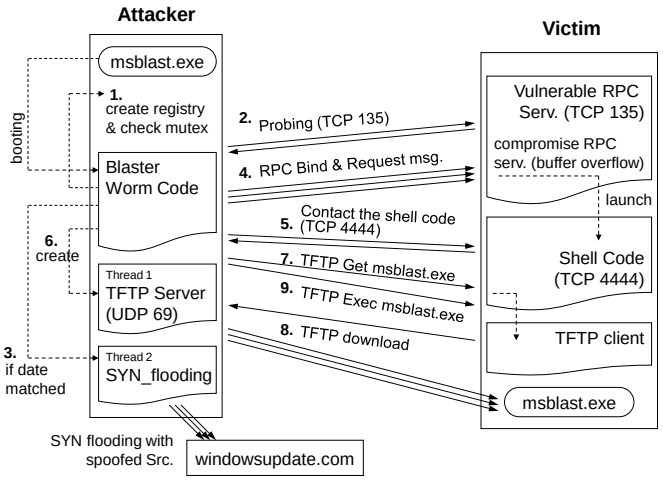

Figure 1. The attack procedure of Blaster.

transport protocol level, there are two TCP connections and one UDP connection. At the service level, there are RPC, TFTP and simple socket-level packet exchanges. More significantly, six attack symptoms are identified: TCP Portsweep, Out of Order RPC Messages, Forced Session Termination, Multiple TCP RSTs, Server Change to Client, and SYN Flooding.

For the example of *Server Change to Client*, it is observed that the victim host, a RPC service provider, changes its role to become a TFTP client connecting back to its former RPC client (i.e., the attacker host) for file downloading. Such role change for a RPC client-server service paradigm is considered unusual. For the *Out of Order RPC Messages*, after sending a RPC BIND to a RPC server, the attacker (a RPC client) doesn't wait for server's BIND ACK but immediately sends a RPC REQUEST back. Such message exchange violates the RPC communication procedure, and is considered as a sign of abnormality. (We will introduce the rest of attack symptoms later.)

Many previous works used simple pattern-matching or protocol inspection techniques. Nowadays, they are not quite effectual [6], [18]. As intrusion becomes more complex, although IDSs like Snort [16] and Bro [17] can provide better detection criteria; however they are rule-based approaches that focus on invariant byte stream, headers, and other correlation criteria. We believe IDSs of the future will need to be capable to detect or infer attacks based on more valuable information from the network properties and characteristics. They are such as the runtime state information of the protocols and services, as well as the relationship between them, and even more — the attack symptom. For the most part, this approach has not been fully exploited in the past. We propose a novel approach that jointly monitor these different levels of activities and infer on these information collectively for effective early detection of stealthy, unknown attacks, or any anomalies on the network.

As many protocols, services and network behaviors are stateful, in this paper, each of the subjects is modeled by a finite state machine (FSM). We focus on the tracking of the procedural-based behavior of and between these three levels. Identifying suspicious or abnormal activities rely on cross-level correlation based on the characteristics of their behaviors and service domain knowledge. Individual packets or connections may seem innocent, but they may cause an attack as a whole.

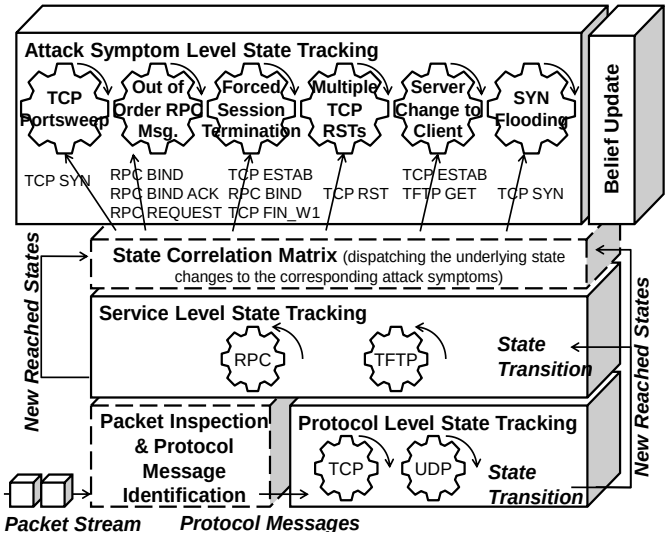

Figure 2. The three-level procedural-based behavior tracking (Blaster).

However, due to the inherent complexities of network protocols and services, realizing a practical behavior-based detection system raises several challenges:

- First, services and also their vulnerabilities do affect different layers in the protocol stack. We should capture behaviors at *multiple* levels and do so efficiently.
- Second, attacks often only manifest themselves as a combination of events that occur *across* multiple layers, and they may not be observed at individual layers.
- Third, to be practical, the system should have a low false positive rate, and provide meaningful *assessments* of anomalies to system administrators.

In the next section, we present the system architecture of the proposed system, Security Monitor (SecMon), and its three-level stateful monitoring. In Section 3, we describe the probabilistic inference model for belief update at SecMon. In Section 4, we illustrate each module of the prototype system. In Section 5, we compare our system with two other detection techniques using various attacks. Finally, we conclude our work.

II. Distributed Security Monitoring

A. Architechture

We consider a coordinated architecture for network anomaly detection. A number of network security monitors, SecMon, are deployed over the network. Each SecMon is responsible for monitoring one or more links. It inspects every network packet by extracting relevant data from the header and payload and keeping track of the states of the protocols and services. It then summarizes and updates the assessments of whether there are any sign of anomalies (i.e., attack symptoms). According to the proposed cross-level inference model and the observed attack symptoms, if the belief score exceeds certain level, SecMon will send an alert along with the anomaly for further action.

B. Multilevel Stateful Network Monitoring

We take the approach of analyzing packet streams by monitoring the states of communication protocols and services, and

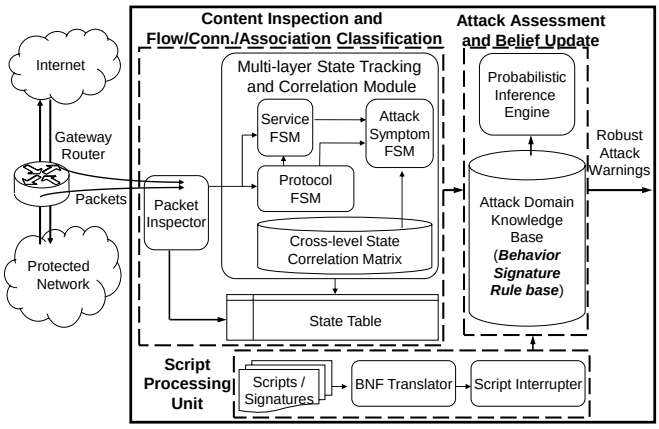

Figure 3. Security Monitor (SecMon) system architecture.

identifying possible attack symptoms. Fig. 2 shows an example of cross-level behavior tracking and inference model. All subjects at each level are described by FSMs. The protocol FSMs are responsible for keeping tracking of the communication contexts. These service FSMs monitor the interactions between client and potential vulnerable server. The information carrying in each packet is extracted and is used to transit the corresponding protocol and/or service FSMs for later use.

As shown in Fig. 1, the detailed behavior between Blaster and victim host is described as follows. Upon infected, the Blaster attacker first creates a registry at the local host. It then sends out probing messages via TCP 135 to find vulnerable victims. If succeeded, the worm code sends the packets carrying exploit code that can compromise the vulnerable hosts. Once compromised, the attacker executes a program at the victim host. It spawns a new process at the victim host waiting for further connection from attacker to the opened TCP port 4444. Meanwhile, the attacker activates a TFTP server and uses TFTP Get command to instruct the victim to launch a TFTP client to download a malicious executable (msblast.exe). Finally, the attack host sends a command to execute the malicious code and starts a new round of spreading. All above protocol and service communications are statefully monitored by FSMs.

The rest of four attack symptoms are described as following. For *Forced Session Termination*, a TCP application usually sends out the last application layer's message before TCP FIN. In the case of Blaster, the victim (i.e., the server) cannot respond due to being buffer overrun, so we can observe that the attacker then forcedly terminates the TCP connection before RPC termination. For *Multiple TCP RSTs*, since the attacker sends the TCP FIN packet immediately to terminate the connection, the late RPC BIND ACK sent by the victim is treated as an abnormal packet at TCP level, so that several TCP RST packets are exchanged between them. *TCP Portsweep* and *SYN Flooding* are commonly seen malicious phenomena.

C. Cross-Level Event Correlation and Attack Tracking

As the state transition of individual protocol or service FSMs may seem individually normal, as explained previously, when we examine them collectively, some anomalies can be detected. Hence, the state transition of attack symptom FSMs should consider all the underlying protocol or service FSMs. Such a cross-level correlation and state transition relationships are specified by the cross-level *State Correlation Matrix* (see Fig. 2). For the example of Blaster, the RPC BIND message triggers a state transition of RPC FSM. According to the cross-level State Correlation Matrix, such a RPC FSM transition maps to the creation of two events, one for Out of Order RPC Messages and one for Forced Session Termination, for further state transition. The tracking of an upper-level FSM incorporates the behavior of its underlying FSMs. An attack symptom represents some direct series of results of the associated underlying services and/or protocols. Thus, it has the memory of the states of the associated services and protocols.

D. Attack Assessment

For anomaly assessment, each SecMon will maintain a list of belief about possible attacks according to current observed attack symptoms. Belief is a probability that assesses the likelihood that the attack is taking place. For the detection of unknown attack, all possible attack symptoms are considered. The proposed probabilistic inference model keeps track of the sequence of attack symptoms observed and updates the beliefs accordingly. Take the example of Blaster, after observing RPC BIND and REQUEST messages, the Out of Order RPC Messages attack symptom FSM reaches to its final state, meaning the attack symptom is observed. Then we perform belief update.

E. System Architechture

Fig. 3 shows the system architecture and the data and control flows of SecMon. The *packet inspector* examines packet header and payload and extracts the fields subject to inspection. Packets are classified into *flows* based on five fields: source IP, destination IP, protocol, source port and destination port. A *connection* (e.g., TCP) is comprised of two flows, one for each direction. The extracted data is fed into the FSM module for state tracking at the three levels (as shown in Fig. 2). The extracted data may be fed to the corresponding transport protocol FSM, service FSM, or both. Upon observing a complete attack symptom, the corresponding belief is updated by the attack assessment and belief update module. All the knowledge and parameters are described in the script, including the protocol, service and the attack symptom FSMs, the probabilities, and the cross-level State Correlation Matrix.

III. THE INFERENCE MODEL

In this section, we present the probabilistic inference and belief update model. A complex attack may undergo a series of activities. One must consider both the pieces of evidence and their sequence. The inference model is to compute and gradually update the belief of whether an attack is taking place based on the sequences of symptom observed.

A. Tracking of Protocols, Services, and Attack Symptoms

All of the protocols and services subject to monitor are modeled by FSMs. They are denoted as $PS_i = <\sigma_i, s_i, \delta_i, t_i, f_i>$. The definition of states (s_i), initial state (t_i), final states (f_i), messages (σ_i) and transitions (δ_i) are based on the network standards or real-world implementations.

A stateful attack symptom is denoted as $Y_j = <\Sigma_j, S_j, \Delta_j(C), I_j, F_j>$.$\Sigma_j$ is the input event set. S_j is a finite set of states. $\Delta_j(C)$ is

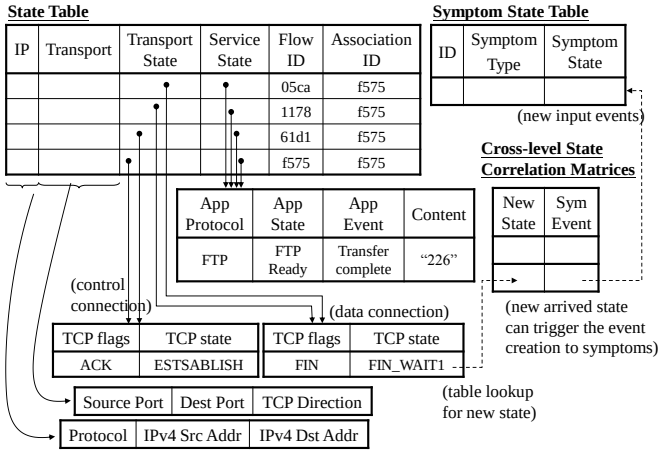

Figure 4. The data structures for cross-level stateful tracking and correlation.

the state transition function with constrains, where $\Delta_j(C) = \{(S_j^u \rightarrow S_j^v, C_j^{u,v})\}$ and $C_j^{u,v}$ is the set of preconditions defined for the state transition from u to v of this attack symptom j. The state transition is successful only when the precondition is satisfied. (For example, in Server Change to Client, while becoming a client, the server should connect back to the *exact* former client, not arbitrary host.) I_j and F_j are the initial and final states set. If a state in F_j is reached, the attack symptom j is observed.

Each attack symptom has a cross-level State Correlation Matrix, $R_{i,j}$, to specify the state relationships between attack symptom Y_j, and other protocols/services FSMs PS_i. The value of elements in $R_{i,j}$ is either null or a two-tuple <state m of PS_i, a list of <event e of S_j>>. Whenever there is a state change at protocol/service FSM, our system creates events according to the list and sends them to the corresponding attack symptoms. An attack symptom has a higher-level semantic meaning of the observable network abnormalities. An expert can design the State Correlation Matrix to realize the conversion between network communication state (of protocol and service FSMs) and semantically meaningful behaviors (i.e., attack symptoms).

B. The Probabilistic Inference Model

In the attack domain knowledge base, an attack A_k is characterized by *a sequence of symptoms*. The inference process works as follows. Let random variables Q_1, Q_2, …, Q_n represent the 1_{st} to n_{th} attack symptom observed in sequence. The assessment of the likelihood (Λ) that how possible attack is taking place is computed as follows:

$$\begin{aligned}\Lambda_n^{A_k} &= P(A_k \mid Q_1, Q_2, \ldots, Q_n) \\ &= \frac{P(Q_1 \mid A_k)\prod_{i=2}^{n} P(Q_i \mid Q_{i-1}, \ldots, Q_1, A_k)}{P(Q_1)\prod_{j=2}^{n} P(Q_j \mid Q_{j-1}, \ldots, Q_1)} P(A_k)\end{aligned} \quad (1)$$

Whenever an attack symptom is observed, the module will update the belief(s) of possible attacks. For the sake of computational traceableness, here, we assume all the conditional probability distributions of the attack and attack symptoms, and between the attack symptoms themselves are Markovian. Namely the probability of the future state depends only upon the present state. We can rewrite (1) to (2).

$$\begin{aligned}\Lambda_n^{A_k} &= \frac{P(Q_1 \mid A_k)\prod_{i=2}^{n} P(Q_i \mid Q_{i-1}, A_k)}{P(Q_1)\prod_{j=2}^{n} P(Q_j \mid Q_{j-1})} P(A_k) \\ &= \begin{cases} \Lambda_{n-1}^{A_k} \dfrac{P(Q_n \mid Q_{n-1}, A_k)}{P(Q_n \mid Q_{n-1})}, & \text{if } n \geq 2 \\ P(Q_1 \mid A_k)\dfrac{P(A_k)}{P(Q_1)}, & \text{if } n = 1 \end{cases}\end{aligned} \quad (2)$$

In (2), the attack assessment is computed incrementally as additional attack symptoms (Q_n) are observed. In this model, the information of $P(Q_1 = Y_x \mid A_k)$, $P(Q_i = Y_y \mid Q_{i-1} = Y_x, A_k)$, $P(Q_i = Y_y \mid Q_{i-1} = Y_x)$ and $P(A_k) / P(Q_1 = Y_x)$ are required. (Let Y_x and Y_y are arbitrary attack symptom.)

C. Determining Probabilities of Attack Symptoms from Known Attacks and Their Variants

The value of $P(Q_1 = Y_x \mid A_k)$, $P(Q_i = Y_y \mid Q_{i-1} = Y_x, A_k)$ for known attacks can be obtained from the past attack traces. For example, for Blaster, we typically observed TCP Portsweep as the first attack symptom; so we have $P(Q_1$ = TCP Portsweep | Blaster) = 1.0. However, there are certain variants of an attack; we can fine tune the probably estimates based on domain knowledge. For instant, we may not observe TCP Portsweep firstly while a stealthy Blaster occurs (see Section 5).

D. A Statistical Approach for Determining Probabilities of and between Attack Symptoms

A statistical approach based on real world traffic traces is used to compute the values of $P(Q_i = Y_y \mid Q_{i-1} = Y_x)$ and $P(A_k) / P(Q_1 = Y_x)$. Assume given a sufficiently long trace and a set of attack symptom, the occurrences of the individual attack symptoms and attacks are counted. The value of $P(A_k) / P(Q_1 = Y_x)$ is computed by counting the relative occurrence of symptom Y_x and attack A_k. Similarly, the $P(Q_i = Y_y \mid Q_{i-1} = Y_x)$ is the probability that symptom Y_y is observed right after the symptom Y_x was observed. See detailed results in Section 5.

IV. Prototype System

The prototype SecMon system is implemented on a PC (Linux kernel 2.6, Pentium 2.4GHz, 512 MB RAM). Our system is inserted at the PREROUTING chain of Netfilter/iptables. Any packet pass by will be processed and inspected first.

A. Content Inspection and Traffic Classification Module

The Content Inspection and Traffic Classification Module assign an identical Flow ID to a packet by a hash function. It helps SecMon to aggregate discrete packets into an abstraction of flow. State Table (see Fig. 4) is a layered pointer-based structure that stores information for different layers and protocols. We support TCP, UDP, ICMP, and many application layer protocols (e.g., HTTP, RPC, FTP, MSN, SMTP, and etc).

We design a more comprehensive relationship above the concept of *flows* and *connections*, named *association*. An association is defined a set of relevant flows which are created by the communicating parties within the same application. Due to

TABLE I. THE CONDITIONAL PROBABILITIES $P(Y_j \mid Y_I)$.

Y_i	Y_j	$P(Y_j \mid Y_i)$
TCP Portsweep	Out of Order Messages	0.000446
Out of Order Messages	Forced Session Termination	0.839687
Forced Session Termination	Multiple TCP RSTs	0.982079
Multiple TCP RSTs	Server Change to Client	0.940810
Server Change to Client	SYN Flooding	0.782119

communication needs, a network application may spawn lots of flows between participating parities. For example, FTP has a control connection and has more than one data connections.

We implement the Aho-Corasick algorithm [21] as our matching module. The processing time of this algorithm has been proved irrelevant to number of keywords. The processing time of string matching is bounded by the length of input packet payload only. We also determine which service a packet belongs to and what the message it is by matching keywords rather than matching port number in the TCP header field.

B. Script Processing Unit and Assessment Parameters

We define a user-friendly, flexible script language in Backus-Naur Form. The script informs our system how to build and transit all the FSMs. A token analyzer and a grammar analyzer of the script language are also developed. For example, when SecMon receive a packet with "220 Ready" message, we consider an FTP is initiated and assign an FTP FSM that updates its state by examining the corresponding flows. Two flows belonging to the same connection share a protocol FSM and flows belonging to same association share one service FSM.

The script language also defines the belief of the attack symptoms and the probabilities used in inference process (including all the probabilities described in Section 3).

V. EVALUATION AND RESULT

A. Statistical Approach for Determining Probabilities

According to a 7-hour traffic collected from the campus core network consisting of 36,000 students, faculties, and staffs, and more than 50,000 hosts. We inject artificial attacks to simulate Blaster. The infection strategy and parameters are based on the Blaster study [20]. One of the hosts in campus network is designated as patient zero. The simulation stops when all of the vulnerable hosts (1% of the hosts) in the campus are identified infected. We compute the assessment of Blaster as follows.

Let $P(Q_1$ = TCP Portsweep | Blaster) = 1.0. P(Blaster) / $P(Q_1$ = TCP Portsweep) is 0.000227, which is obtained by statistical approach from the trace. After observing the first symptom, based on (2) the belief is P(Blaster | Q_1 = TCP Portsweep) = 0.000227. From statistical analysis, the $P(Q_i$ = Out of Order RPC Messages | Q_{i-1} = TCP Portsweep) is 0.8396. If Out of Order RPC Messages is then observed, then the belief is raised up to 0.5079. For each additional symptom observed — Forced Session Termination, Multiple TCP RSTs, Server Change to Client and SYN Flooding — the belief of Blaster attack rises to 0.6049, 0.6160, 0.6547 and 0.8371, respectively. The subsequent observations of more symptoms reinforce the assessment to assure the alert raised is accurate. Table I show the conditional probabilities calculated from the 7-hour traffic trace (with Blaster injected). However, the probabilities are base on the selected environment and historical traffic data. It may take training and testing technique to fine tune the confidence level of different network, which is beyond the scope of this works.

B. Detecting Known Attacks

We use Snort to represent the signature-based IDS, and use Bro as an anomaly detection system. We replay the same Blaster attack traffic trace to all the comparing systems. For the signature-based system, four alerts are raised: a "priority 1" alert for detecting overflow attempt by payload signature at 1.8 second, a warning of retrieving malicious executable file (msblast.exe) at 2.3 second, a "priority 2" warning of TFTP Get command at 2.3 second, and a "priority 3" TCP Portsweep at 2.9 second. The anomaly detection mechanism output an alert at 81.64 second that an IP has scanned more than 50 hosts.

In SecMon, we can observe five attack symptoms in our traces (SYN Flooding is not observed because it only occurs on specific dates). TCP Portsweep, Out of Order RPC Messages, Forced Session Termination, Multiple TCP RSTs, and Server Change to Client are observed at 0.0015, 1.8035, 1.8036, 1.8039, and 2.3078 second, respectively.

C. Detecting Polymorphic Variants of Known Attacks

For understanding how the different approaches handle polymorphic variants, we encrypt the exploit payload of Blaster. In this case, the signature-based alert is not raised in Snort. But other rules and policies that detect the scanning behavior can raise alerts when a sufficient number of scans are seen. Since polymorphism only changes the appearance of attack packets and does not affect the attack procedure and behavior, the SecMon can still successfully detect all attack symptoms.

D. Detecting Setalthy Scanning Variants

In this experiment, we made a "smart" Blaster that has prior knowledge of the vulnerable hosts/services. Thus, it can directly attack targets without any probing activities. All the scanning detection rules or policies in Snort, Bro and SecMon fail to trigger alerts in this case. Despite this, SecMon still captures the rest of all abnormalities. With only four attack symptoms (Out of Order RPC Messages, Forced Session Termination, Multiple TCP RSTs, and Server Change to Client), the SecMon can still come up with a belief score at 0.4914. Moreover, if an extra SYN Flooding is observed, the score would be 0.6283.

E. Unknown Attacks

We use Sasser [4] to emulate how the systems react to unknown attacks. We explicitly remove all the knowledge about Sasser from Snort, Bro and SecMon, and only retain the prior knowledge based on the Blaster attack. For Snort, a notification of "NETBIOS SMB-DS access" is raised. After outbreak, Snort raised a "TCP Portsweep". For Bro, the scan policy is still workable, even Bro does not have any knowledge about Sasser. For SecMon, it still observes three attack symptoms. SecMon is based on the knowledge of attack symptoms exhibited in previously known attacks; it has certain potential to detect unknown attacks. The more attack symptoms the SecMon has, the more detection capability it has to detect un-

TABLE II. THE NUMBER OF OCCURANCE AND PROCESSING TIME OF ATTACK SYMPTOMS FOUND IN A 7-HOUR CAMPUS TRACE.

Attack Symptom	# of occurrences	Processing Time (sec)
TCP Portsweep	289,476	444
Out of Order Messages	1,536	19
Forced Session Termination	81	20
Multiple TCP RSTs	271	625
Server Change to Client	0	1,137
SYN Flooding	717,170	379

known attacks. The attack symptoms are sometimes shared by different attacks due to program and vulnerability characteristics. For example, buffer overrun usually causes anomaly protocol execution due to not being able to respond messages. We anticipate that an attack symptom can be identified in the runtime, once we well describe the normal execution model of protocols/services and the relationship between them from network specifications or experts' experiences. Although, currently known attacks can give us good clues to generate good example of attack symptoms, yet we are now investigating the problem of automatic attack symptom generation.

F. Detection Accuracy

In SecMon, an alert is not based on a single attack symptom, but rather on progressive inference and correlation of a sequence of attack symptoms. Table II shows the number of occurrences of the attack symptoms of Blaster from a seven-hour traffic trace with more than 45 million flows and 680 million raw packets (within a 378GB trace file) of a campus network (having no Blaster attacks). The frequency of different attack symptoms shows that they possess different meanings (weights) while accessing an attack. That is why we introduce the probabilistic inference model to correlate them. We also calculate consecutive observations of two attack symptoms in this Blaster-free trace, and only 37 instances (false positives) are found. It gives us good confidence that progressive inferences and symptom correlation should have low false positive rates. However, it does not prove SecMon is false positive-free.

G. Computation Performance

SecMon takes time to identify every attack symptom in the traffic. We simply measure the total processing time identifying attack symptoms in Table II. The processing time for all the attack symptoms is roughly less than 10% of the entire duration (7-hour). In other words, SecMon roughly has 10% overhead. We also calculate the performance degradation for a gateway with and without SecMon. For TCP and UDP tracking, the overheads incurred are about 26.8% (from 261 to 191.4 Mbps) and 9% (from 312.5 to 284.2 Mbps), respectively. However, it can be improved with hardware support.

VI. CONCLUSION

As attacks continue to evolve, the need for detection mechanisms that are robust to attack-evasion strategies is imminent. In this context, our approach is based on the key insight that attacks follow common procedures and exhibit anomaly invariant behaviors (attack symptom). In this paper, we addressed several key challenges in the design of such behavior-based detection system: (1) implementing an efficient mechanism for capturing behavioral features using FSM representations, (2) capturing attack symptoms using cross-level correlation, (3) presenting a meaningful assessment of anomalous observations using a probabilistic inference model, (4) presenting a flexible framework and script language that are able to extend the protocol, service and attack symptoms.

Our evaluations establish that the SecMon approach is robust to several common attack-evasion strategies including polymorphic, stealthy variants, and previously unknown attacks. We also show that using the cross-level correlation in conjunction with the probabilistic inference model significantly reduces the effective false positive rate, by considering a sequence of anomalous observations collectively rather than in isolation. These results confirm our approach is a robust, practical, promising alternative to attack detection.

REFERENCES

[1] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time", Proc. USENIX Security, 2002, pp.149-167.

[2] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A Taxonomy of Computer Worms," Proc. Workshop on Rapid Malcode, 2003.

[3] CERT. (2003, Aug.). CERT Advisory CA-2003-20 W32/Blaster worm. [Online]. Available: http://www.cert.org/advisories/CA-2003-20.html

[4] Microsoft. (2006). Sasser. [Online]. Available: http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?name=Win32/Sasser

[5] CAIDA. (2001). CAIDA Analysis of Code-Red. [Online]. Available: http://www.caida.org/research/security/code-red/

[6] M. Allman, E. Blanton, V. Paxson, and S. Shenker, "Fighting Coordinated Attackers with Cross-Organizational Information Sharing," Proc. fifth Workshop on Hot Topics in Networks (HotNets-V), 2006.

[7] A. Lakhina, M. Crovella, and C. Diot, "Characterization of Network-Wide Anomalies in Traffic Flows," Proc. 4th ACM SIGCOMM conference on Internet measurement, 2004, pp. 201-206.

[8] A. Wagner, and B. Plattner, "Entropy Based Worm and Anomaly Detection in Fast IP Networks," Proc. IEEE International Workshops on Enabling Technologies (WET ICE), 2005, pp. 172-177.

[9] C. C. Zou, W. Gong, D. Toesley, and L. Goa, "The Monitoring and Early Detection of Internet Worms," Transaction on Networking, 2005.

[10] D. Moore, C. Shannon, G. M. Voelker, and S. Savage. "Network telescopes: Technical report," CAIDA, Tech. Rep. TR-2004-04, 2004.

[11] S. Chen, X. Wang, L. Liu, X. Zhang, and Z. Zhang, "WormTerminator: An Effective Containment of Unknown and Polymorphic Fast Spreading Worms," Proc. ACM/IEEE ANCS, 2006, pp. 173-182.

[12] D. Wagner, and D. Dean, "Intrusion Detection via Static Analysis," Proc. of IEEE Symposium of Security and Privacy, pp. 156-168, 2001.

[13] CERT Coordination Center: Overview of Attack Trends. [Online]. Available: http://www.cert.org/archive/pdf/attack_trends.pdf

[14] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," Proc. IEEE Symposium on Security and Privacy, 2005, pp. 226-241.

[15] W32.Welchia.Worm. Available: http://www.symantec.com/security_response/writeup.jsp?docid=2003-081815-2308-99

[16] M. Roesch, "Snort – Lightweight Intrusion Detection for Networks," Proc. 13th Systems Administration Conference (LISA '99), 1999.

[17] V. Paxson, "Bro: A system for Detecting Network Intruders in Real-Time," Computer Networks, vol. 31, pp. 2435-2463, 1999.

[18] T. Karagiannis, D. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," Proc. the conference on applications, technologies, architectures, and protocols for computer communications, pp. 229-240, 2005.

[19] Trend Micro. (2003, Aug.). MSBLAST. [Online]. Available: http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM