Behavior Profiling for Robust Anomaly 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
Institute of Information Science
Academia Sinica
Taipei, Taiwan
mcc@iis.sinica.edu.tw

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

***Abstract*—Internet attacks are evolving using evasion techniques such as polymorphism and stealth scanning. Conventional detection systems using signature-based and/or rule-based anomaly detection techniques no longer suffice. It is difficult to predict what form the next malware attack will take and these pose a great challenge to the design of a robust intrusion detection system. We focus on the anomalous behavioral characteristics between attack and victim when they undergo sequences of compromising actions and that are inherent to the classes of vulnerability-exploit attacks. A new approach, Gestalt, is proposed to statefully capture and monitor activities between hosts and progressively assess possible network anomalies by multilevel behavior tracking, cross-level triggering and correlation, and a probabilistic inference model is proposed for intrusion assessment and detection. Such multilevel design provides a collective perspective to reveal more anomalies than individual levels. We show that Gestalt is robust and effective in detecting polymorphic, stealthy variants of known attacks.**

Anomaly detection; attack accessment; behavioral analysis; finite state machine; netwrok service

I. Introduction

Internet attacks involve continuously evolving evasion techniques, such as polymorphism and stealthy scanning. Consequently, conventional detection systems that use a combination of signature-based and/or rule-based anomaly detection techniques no longer offer sufficient protection. Moreover, it is difficult to predict what form or strategy the next malware attack will take, which poses a great challenge to the design of a robust intrusion detection system. In this paper, behavior profiling has been proposed as a means of identifying malware attack. In traditional criminal investigations, profiling refers to the use of specific characteristics, such as gender and age, to make generalizations about a person to identify that he or she may be engaged in an illegal activity [1]. In cyberspace security, the key issue in profiling is how to identify the distinctive characteristics of attacks that are robust and can be manipulated to prevent further intrusions, especially stealthy and unknown attacks.

In the literature [2], intrusion detection techniques were generally classified into different approaches. Most works on profiling or fingerprinting focused on content-based signatures, which capture an attack's characteristics by deriving the most representative content sequence from the attack packets. Signature-based techniques are most commonly used for the detection of known attacks (e.g., Snort [3]). However, these methods cannot detect unknown attacks, and they may not be effective for variations of known attacks like polymorphic attacks. In addition, automatically generating and distributing signatures (e.g., EarlyBird and Polygraph in [2]) for new attacks remains an open research issue.

Anomaly detection overcomes the limitation of signature-based detection techniques by focusing on normal behavior. Typically, normal behavior are observed and characterized to find the presence of attacks. Machine learning and statistical analysis techniques are often used to create a profile of the normal behavior, and any deviation from the normal behavior profile is flagged as a potential attacks. In the past, most network-based anomaly detection works focused on profiling contact behavior to detect network abnormalities, e.g., a large volume of traffic generated by scanning, the high rate of failure connection attempts, or the epidemic spreading phenomenon of a worm infection [4]-[6]. There are also works focus on host-based program execution profiling [19]. Different from them, we focus on network service execution behavior as a basis to profile a network activity.

While current detection methods are only effective for specific attacks and certain assumptions under consideration (e.g., a tree-structure in rapidly spreading worm attacks), future sophisticated attacks could try to evade them. For example, attacks may adopt stealthy probing and scanning strategies to evade traffic-based detection techniques [7]. Polymorphic attacks that mutate to change the payload across attack instances can evade signature-based detection techniques. Detection techniques also face the challenge of detecting new, unknown attacks. Clearly, we need robust and effective detection techniques against future attacks.

A number of works (e.g., [2], [8], and [9]) have demonstrated the effectiveness of using behavioral patterns to detect classes of unseen worms that possess epidemic propagation characteristics. In [8], the authors studied the inter-machine propagation pattern exhibited by worms and identified three behavior patterns: "Server Changes to Client", "α-in-α-out", and an inductive relationship. We consider these relationship or epidemic propagation could be easily evaded by using certain sophisticated, stealthy techniques, such as those described in [7]. In [9], a unique temporal sequence of message exchange is used as a behavioral footprint to detect known worms that repeats an identical attack procedure as they propagate. Although these methods may detect polymorphic attacks, they are only suitable for detecting known

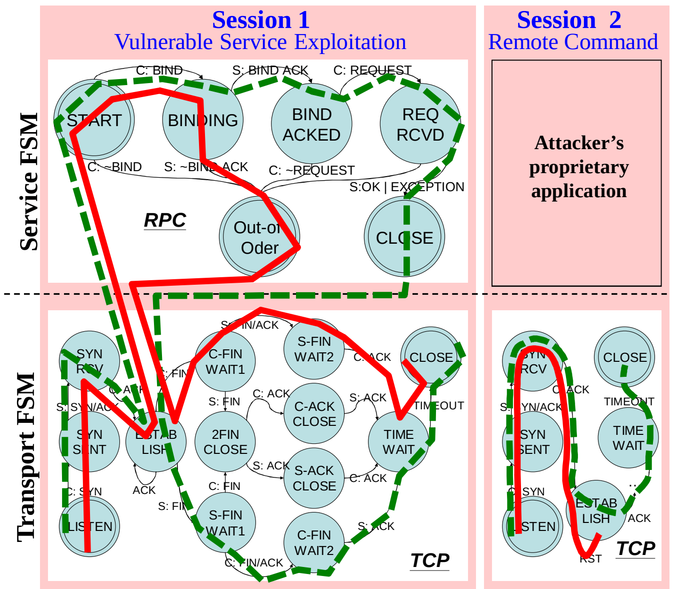

Figure 1. The execution paths of Blaster.

attacks. Moreover, these behavior patterns are mostly manually constructed by security experts, such as in [10], [13], and [17], and developing high quality pattern is not easy.

To confront the future unknown attacks, we propose a profiling mechanism to capture the deviated behavior of attacker's underlying protocol executions collectively while it undergoes the vulnerability exploitation. We point out that more anomalies can be revealed by our multilevel (i.e., transport, service, and attack symptom level) design and the proposed probabilistic inference model can decreased the false positive of anomaly-based detection technique.

The rest of the paper is organized as follows. In the next section, we present the concept of behavior profiling and an example of attack Blaster to illustrate our main observation and idea. In Section 3, we present the framework to profile the attack behavior using a multilevel behavior tracking approach. The probabilistic inference model is also proposed in this section. In Section 4, the detail steps of how to construct the multilevel behavior profile is described. In Section 5, we demonstrate the experiment result of our system, Gestalt, with different attacks and their variants. Finally, we conclude our work in the last section.

II. Behavior Profiling

Instead of looking for the characteristic behavioral patterns of an infected host in the scanning or spreading phases, we focus on behavior in *compromising phase*. Specifically, we focus on the anomalous behavioral characteristics exhibited by the attacker and the victim hosts when they undergo sequences of activities to exploit the network service vulnerability. Class attacks, especially for non-human involved attacks, such as worm, botnet, operating fingerprinting [20], and vulnerability probing are in our scope. We refer to these distinctive anomalous behavior characteristics as *behavior signatures*. To avoid detection by a behavior signature, an attack must change its fundamental behavior. However, it is a challenge to modify such vulnerability-exploit behavior.

The execution of a network application or service is essentially a communication between distributed hosts employing certain network protocols and service modules. To progress vulnerability exploitation over the network, an attack often causes problems that disrupt the execution of the vulnerable service. In other words, certain behavior that deviates from the normal execution of the underlying protocols and services (called *deviated behavior* hereafter) can be expected when a server is under attack. Although most existing transport protocols and services have already implemented routines to handle and respond to unexpected inputs, yet there are still certain malicious inputs cannot be correctly handled. The state of service execution may become undetermined and the server process may hang to time out due to unexpected input. For instance, in the class of buffer-overflow attacks, the server process often "hangs" after being buffer overflowed, since it can neither respond to, nor send, messages as a normal server. This also causes the underlying TCP connection to time out. Subsequently, the system kernel undergoes a cleanup of the data structures associated with the connection. Thereafter, any packets exchanged on the connection are considered "non-legitimate" by both the attacker and the victim hosts. Such abnormal sequences of message exchange and the "hanging" situation can be identified by monitoring the execution contexts of the protocols and services.

Fig. 1 shows the deviated execution paths exhibited between the attacker and the victim hosts in Blaster. Three sessions are established in the attack: the RPC session, which exploits the buffer overflow vulnerability of the RPC program; the remote command session, which runs on top of a TCP connection for the attacker to send remote control commands to the victim; and a TFTP session (which is not shown in Fig. 1) initiated by the victim to download a malicious file from the attacker. The state transition diagrams of the protocols and services are depicted in the figure. In each diagram, we show the normal (i.e., most frequent) execution paths and the paths taken by Blaster. Two obvious deviated behavior patterns can be identified in the RPC session. The first occurs when the attacker sends RPC messages in the wrong order, and the second is evident when the attacker terminates the TCP connection while the RPC session is still in a non-CLOSE state. The latter is a forced session termination that can be observed by cross-checking the RPC and TCP executions. These activities are not expected in normal invocations of RPC and TCP. Another uncommon activity is noted in the remote command session when the underlying TCP connection is terminated with an RST flag rather the usual FIN flag. This is also a sign of an alert.

Although finding deviations has been an effective approach for a long time, yet we notice that certain abnormal behavior can *only* be identified when the executions of one or more protocols and services are *collectively* tracked and related. Our design can reveal more behavior signatures than conventional approaches that only focus on individual protocol. In the above example, if we tracked the execution of the

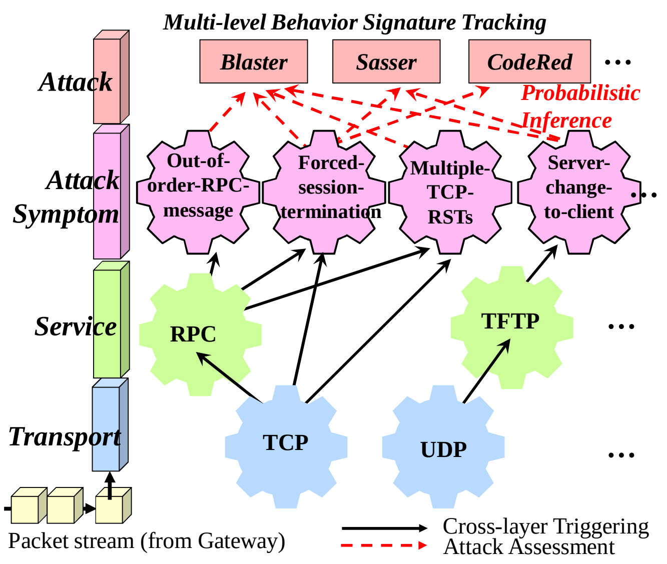

Figure 2. The multi-level finite state machine based network behavior tracking, correlation and analysis model.

TFTP session alone, it would not exhibit any unexpected behavior. However, for an RPC server, if we correlate the activities of RPC and TFTP sessions, it would clearly be unusual if the RPC server established a TFTP session with its RPC client (i.e., the attacker) or with any other hosts. Such behavior is considered as an indication of abnormality. From a transport protocol or service perspective, individual deviated activities may be subtle or normal when observed alone; however, when these deviations are collectively tracked and related; the signs of an attack should stand out.

Based on the observation that almost all vulnerability-exploit attacks exhibit one or more distinctive deviated behavior patterns or *attack symptoms* to progress vulnerability exploitation, we propose an attack symptom tracking architecture and attack profiling method. Through observations of a sufficient number of attack symptoms, the intrusion detection system than infers the likelihood of possible attacks.

We believe that behavior-based intrusion detection techniques will be effective for coping with the challenges of detecting future stealthy and new attacks. The advantage of being able to identify the behavior signatures exhibited by different classes of attacks is that it is not necessary to know the specifics of an attack *a priori*. This is important because evading a behavior signature requires a fundamental behavior change in an attack's service invocation and vulnerability exploitation, not just its network footprint. Changing fundamental attack behavior would be much more challenging.

A number of works have modeled the vulnerability behavior of host-based intrusion detection. In [10], Shield uses state machines to define vulnerability-specific signatures by payload characteristics that lead to any remote exploits of the vulnerability. For unknown vulnerability attack detection, a follow-up work, ShieldGen [11] generates signatures by analyzing zero-day attacks, constructing new potential attack instances (probes) and testing the probes in virtual machines. Most of the time, the vulnerability needs to be well-studied, and the result cannot be used by other vulnerabilities. In [18], it demonstrates the theory and an automatic vulnerability signature generation algorithm with a sample exploit.

III. Intrusion Detection by Attack Symptoms

Adopting the attack symptom property, we propose an attack symptom tracking system for robust intrusion detection that is based on a multilevel, finite-state machine (FSM).

A. Multilevel Behavior Tracking and Correlation

Most methods use simple events or statistics to describe abnormal behavior [4] [9] [12]. However, in the Blaster attack shown in Fig. 1, certain deviated behavior can *only* be recognized by collectively observing activities between the attacker and the victim hosts. We observe that attacks usually involve several aspects of activities in the communication protocol, service/application and the service model (e.g., client-server or P2P). In the proposed system, the security status of communications is monitored by finite state machine on three levels: transport, service, and attack symptom. Fig. 2 shows the architecture of our multilevel inference model for behavior tracking and correlation, and the progressive inference model. The FSMs at the transport and service levels are designed to track both normal and abnormal behaviors in the transport and service states of network connections and service sessions. The top-level activity tracking is accomplished by attack symptom FSMs, which model high-level abstractions of abnormal behavior. The state transitions of the attack symptom FSMs are based on the observations and correlations of the states in the lower two levels.

Attacks are assessed based on observations of the attack symptoms exhibited by communicating hosts. Three logical entities, *flows*, *connections*, and *associations* are used to monitor and correlate the communication relationships between hosts. Packets are first classified into unidirectional *flows* based on a 5-tuple: source IP, destination IP, protocol, source port, and destination port. A *connection* refers to two unidirectional flows with the same protocol, but with the source and destination addresses and ports interchanged, e.g., a TCP connection consists of two unidirectional flows. An *association* is a set of relevant flows established in the context of an instance of a network service. The classification of flows into associations is based on knowledge of the services, e.g., an FTP application may have one control connection and zero or more data connections.

Some works use FSMs to monitor network activities. NetSTAT [13] employs state transition diagrams to model general network state changes or events (e.g., active connections, interactions states, network topology and configurations) to represent known attack scenarios. However, the authors need to specify the state transition diagrams based on their experience for effective intrusion detection. In some intrusion detection systems, FSMs are also used to record the state of the network to enhance intrusion detection. For example, in Bro [14], some policies incorporate state information to specify throttling rules on traffic flows, rules for alerting security-related events, and signatures for known attacks and vulnerabilities. In contrast, we use the state machine techniques to identify distinctive behavioral deviations exhibited during the compromising phase of attacks. We explain how we construct these FSMs later in the article.

B. Attack Assessment

Each attack symptom reflects a particular distinctive anomalous behavior pattern exhibited by some malware attacks. The assessment of attacks progresses as more attack symptoms are observed. We use a probabilistic inference model to infer and compute the belief score of possible attacks, known or unknown, based on observations of the attack symptoms. There are two important reasons for employing the model. First, each attack, simple or sophisticated, will exhibit one or more attack symptoms. Each attack symptom has a different degree of significance in the attack evaluation. Some attack symptoms are considered more important than others, depending on the role of the corresponding activity executed in the attack, and whether it could be avoided, or modified. Second, since future attacks are likely to become stealthier in order to outwit current intrusion detection systems, it is important that detection of future unknown attacks should not rely on single or individual observations. Hence, each attack symptom is associated with a probability value to represent its anomaly level in attacks.

Conditional joint probabilities are adopted to calculate the aggregate probability when a number of attack symptoms are observed. Thus, the model provides network administrators with a meaningful way to assess information about the confidence scores of suspected attacks. Let random variables Sq_1, Sq_2, …, Sq_n represent the 1_{st} to n_{th} attack symptom observed. The assessment of the likelihood (Λ) that how possible attack is taking place is computed as follows:

$$\Lambda_n^{A_k} = P(A_k \mid Sq_1, Sq_2, \ldots, Sq_n) = \frac{P(Sq_1 \mid A_k)\prod_{i=2}^{n} P(Sq_i \mid Sq_{i-1}, \ldots, Sq_1, A_k)}{P(Sq_1)\prod_{j=2}^{n} P(Sq_j \mid Sq_{j-1}, \ldots, Sq_1)} P(A_k) \quad (1)$$

For the simplicity of computing conditional probability table, we assume the observation of attack symptom is Markovian. Namely the probability of the next attack symptom depends only upon the present one. If we take this Markovian assumption, we can rewrite (1) to (2). Although such an assumption may cause the value of Λ be overestimated or underestimated when n > 2, yet maintaining the conditional probability table of (1) may not be practical. In practice, we can combine related behavior into one attack symptom to make sure the behavior of each attack symptom is as independent as possible to mitigate the problem of overestimation and underestimation.

$$\Lambda_n^{A_k} = \frac{P(Sq_1 \mid A_k)\prod_{i=2}^{n} P(Sq_i \mid Sq_{i-1}, A_k)}{P(Sq_1)\prod_{j=2}^{n} P(Sq_j \mid Sq_{j-1})} P(A_k) \quad (2)$$

In this inference model, the information of $P(Sq_1 = S_x \mid A_k)$, $P(Sq_i = S_y \mid Sq_{i-1} = S_x, A_k)$, $P(Sq_i = S_y \mid Sq_{i-1} = S_x)$ and $P(A_k)$ / $P(Sq_1 = S_x)$ are required in prior (where S_x and S_y are attack symptoms). A statistical approach based on real world traffic traces is used to compute these values. The threshold of Λ depends on network configuration. In this paper, we do not specifically set a fix value for Λ; rather we observe the increment of Λ when each attack symptom is observed. The value of Λ is computed and shown in evaluation section. The details about probability operations, quantifying conditional probabilities, and probability independency issues can be found in probability reasoning textbook, such as [21].

IV. Profiling Attacks

The construction of attack symptom FSM involves the following four steps.

1) Construct Protocol and Service FSMs

Since the states of attack symptoms represent temporal and semantic summarization of the anomalies tracked by the service and transport FSMs, we first construct protocol and service FSMs based on the standard specifications and/or the de facto implementations. Each FSM models the interaction procedures of the associated protocol/service between communicating hosts, and the definitions and formats of the messages exchanged. It is particularly imperative to model what action should be taken when these FSMs receive unexpected packets so to identify possible deviated activities in the associated protocol or service. For service-level FSMs, the communication model, such as simple one-tier client-server, peer-to-peer, three-tier client-server communication paradigm, is also taken into account. This is crucial for describing the characteristic interaction behavior between communicating hosts as well. The other essential features modeled include: the number of connections established, who initiates each connection, whether fixed or random ports are used, and the time intervals between the connections.

We simply manually construct these FSMs. Take TCP [22] for example, we first identify all possible messages in this protocol based on the TCP Header Format. For TCP, the 6-bit Control Bits are used to differentiate different messages. We also add a message called “unknown”; in case we receive a message that is not defined in the protocol specification. Then we identify the states of this protocol based on the TCP Connection State Diagram provided in [22]. Then, the transitions are linked between these states. We also add several additional states for error handling mechanism so that the FSM can transit to one of the error state if unexpected messages or undefined messages are received. All protocol and service FSMs are constructed in such way.

The construction of these FSMs can be a training-testing approach by collecting real-world traffic traces to build the model of FSM. We are now investigating a semi-automatic approach to generate protocol and service FSMs. However, such approach is beyond this paper’s scope.

2) Collect Complete Attack Traces

The goal of this step is to discover the tactics and methods (procedures) used in an attack, and identify the distinct abnormal activities that are crucial and unique to the attack. Note that a complete trace of raw packets exchanged between the attacker and the victim(s) is necessary to gain a complete understanding of the “communication context” (both normal and abnormal) of an attack at the transport and

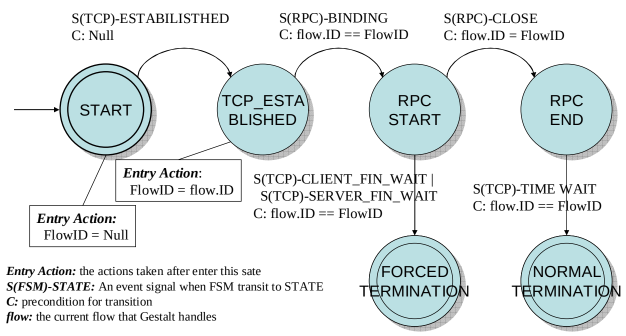

Figure 3. "Forced session termination" attack symptom FSM.

service levels. This is because certain clues about abnormal signature (attack) behavior are hidden in the traces, which are not usually reported in attack whitepapers. Examples of such clues includes 1) what additional connections have been created; 2) why, in addition to the targeted vulnerable service session, the attacker needs another connection to complete the exploitation (e.g., downloading malware code); and 3) what unexpected activities and packet exchanges have occurred. The information is important for understanding and analyzing deviated activities in previous attacks.

3) Find Deviated Execution Paths

Attack traces are fed to the corresponding transport and service FSMs to analyze deviated execution paths and determine how frequently they occur. One way to perform this task is to use the training and testing techniques. We first calculate the occurrence frequency of each state and transition path by training the system with packet traces that do not contain attacks. Then, we use the attack traces to compare the occurrence frequencies with those derived in the training set. As a result, the distinctive deviated execution paths taken by attacks can be identified. (In [21], it also describes the details of counting process.)

4) Construct Attack Symptoms

After collecting the deviated execution paths in the protocol and service FSMs identified for each attack, we can construct attack symptom FSMs. Since attack symptoms represent a temporal, semantic summarization of the anomalies tracked by the service and transport FSMs, the state transitions of attack symptom FSMs are driven by the transitions of one or more protocol and/or service FSMs. Every (deviated and normal) state transition of the underlying FSMs is extracted as an event of upper layer attack symptom FSM. In the meantime, a temporal state of the attack symptom FSM is also created. We then concatenate all new events and states along with the time that they occur to construct the attack symptom FSM. Such an attack symptom FSM is the profile of the attack. Note that human expert may need to split the entire attack behavior into smaller pieces by their time of occurrence. Fig. 3 shows one of the attack symptom of Blaster, named "Forced session termination", whose state transitions are driven by the events occurred and the states reached of the TCP and RPC. At this time, a network expert can manually fine-tune the attack symptom FSM by removing some events or states that can considered irrelevant to the attack. For example, TCP three-way hand sharking process may be irrelevant to the Blaster attack so that we can replace the three-state process by a state "TCP ESTABLISHED" in the attack symptom FSM or we can even remove it.

Then a *State Correlation Matrix* ***R*** is used to describe the relationship between transport/service FSMs and attack symptom FSMs. The value of the element $R_{i,j}$ is either null or a two-tuple <state *m* of TS_i, a list of <event *e* of S_j>>, which indicates that the reach of state *m* of TS_i triggers the creation of (one or more) event *e* to the attack symptom S_j. This matrix specifies how attack symptoms transit based on the state changes of transport and service FSMs. An example is $R_{i,j}$ = <state "BINDING" of RPC FSM, event "RPC-BINDING" of "Forced session termination">), which means the reach of state "BINDING" of RPC FSM will trigger the creation of an event "RPC-BINDING" to the "Forced session termination" attack symptom.

V. Performance Evaluation

We implemented a prototype system, called *Gestalt*, on a PC-based Linux platform. The name "Gestalt" is used to emphasize the "wholeness" principle in robust detection of future vulnerability-exploit attacks. The objective is to deploy the system at the gateway router of a network to capture packets between the protected network and the Internet.

In the first experiment, we analyzed 6 attack traces using the proposed attack symptom profiling method. As we mentioned in previous section, the complete attack trace is needed to capture the behavior of the attack so we collect the execution files of the malware from the Internet and reproduce these attacks in the laboratory. Although there are some tools, such as honyd [23], for collecting the execution file of the attack, it is still not easy to collect unknown malware and non-alive malware form the Internet. Unknown attacks may not be identified by these tools, and non-alive malware can only be found in the laboratory or archive, which may not be public available or the execution file is modified to be harmless. In this experiment, we collect six execution files of the malware, and then execute them in a controlled environment, log the packets exchanged between attacker and victims, and study the malware behavior to ensure all the attack behavior are logged. Table I presents the results. Although these attacks were discovered in different years and targeted different service vulnerabilities, they exhibited and shared certain characteristic abnormal behavior, because they adopted common attack methods and procedures. Usually, attacks have three phases: probing, compromising, and propagating. As [15] mentioned, the probing and propagating phases are easily disguised. However, in order to control the victim, exploiting the vulnerability is necessary. In Table I, we can see there are certain common consequences after/while exploiting a service, such as abnormal message order, error/incorrect format message, and abnormal state transitions. They are the entry point that attacker enter the victim. Usually, the compromised server does not know how to normally reply the malicious request, thus "Forced session termination" often occurs. The service timeout mechanism of TCP would forcedly terminate the no-response service. Furthermore, exploited vulnerable servers may need another mali-

TABLE I. THE ATTACK SYMPTOMS FOUND IN ATTACK BEHAVIOR PROFILING

Phase	Symptom	CodeRed	CodeRed II	Nimda	Blaster	Welchia	Sasser
Probing	**TCP-portsweep**	TCP port 80	TCP port 80	TCP port 80	TCP port 135	Broadcast PING[b]	TCP port 445
Com-promis-ing	**Out-of-order-message**[a]	HTTP message	Incorrect HTTP format	HTTP error message	RPC message	No	No
	Forced-sess.-termination	HTTP session	HTTP session	No	RPC session	RPC session	SMB-RPC session
	Multiple-RSTs	No	No	No	Yes	No	Yes
	Server-change-to-client	No	No	HTTP server to TFTP client	RPC server to TFTP client	RPC server to TFTP client	SMB-RPC server to FTP client
Propa-gating	**Flooding**	HTTP flooding	No	No	TCP SYN flooding	No	No

a. "Out-of-order-message" includes abnormal message order, error/incorrect format, and abnormal state transitions. b. The compromised victim searches next target by sending ICMP requests to random hosts

TABLE II. THE DETECTION RESULT OF DIFFERENT DETECTION TECHNIQUES ON DIFFERENT ATTACK VARIANTS[c]

Attack Variants	Gestalt (behavior based)			Content-signature-based	Anomaly-based
	Attack Symptom	*Belief Score*	*Belief Score if B_6 is observed*		
Polymorphic Variant	B_1, B_2, B_3, B_4, B_5	0.6547	0.8371	C_3, C_4	A_1
Slow Scan Variant	B_2, B_1, B_3, B_4, B_5	0.6296	0.8050	C_1, C_2, C_3, C_4	A_1
Selective Scan Variant	B_2, B_3, B_4, B_5	0.4914	0.6283	C_1, C_2, C_3	
Unknown Attack	B_3, B_4, B_5	0.7124	0.9109	C_4, C_5	A_1

c. B_1: TCP-portsweep; B_2: Out-of-order-RPC-message; B_3: Forced-session-termination; B_4: Multiple-TCP-RSTs; B_5: Server-change-to-client; B_6: Flooding; C_1: Snort priority 1 alert, sid #2315 – overflow attempt; C_2: Warning – malware download; C_3: Snort priority 2 warning, sid #1444 – TFTP; C_4: Snort priority 3 notification – scan; C_5: Snort warning – SMB; A_1: Bro policy – scanned IPs > threshold.

TABLE III. THE OCCURRENCE OF INDIVIDUAL AND CONSECUTIVE ATTACK SYMPTOMS OF BLASTER IN A 7-HOUR TRAFFIC TRACE

Number of Attack Symptom Occurrences			
Individual		***Consecutive***	
TCP-portsweep (TP)	289,476	TP+OR	31
Out-of-order-RPC-message (OR)	1536	OR+FT	3
Forced-session-termination (FT)	81	FT+MR	3
Multiple-TCP-RSTs (MR)	271	MR+SC	0
Server-change-to-client (SC)	0	SC+SF	0
SYN-flooding (SF)	717,170		

cious executable file for latter use, so they may act like a client to download file from the attacker. These attack symptoms are captured in our experiment.

In the second experiment, we conducted preliminary experiments to determinate how different intrusion detection approaches handle stealthy and unknown attacks. Specifically, we compared Gestalt with the content-signature-based and anomaly-based techniques implemented in Snort and Bro, two popular intrusion detection systems. We modify Blaster source code to generate its polymorphic and two stealthy variants (slow scan and selective scan) to demonstrate the detection capability of each detection approaches. For unknown attacks, we used Sasser, a buffer overflow attack. We explicitly removed the knowledge about Sasser and retained all the other rules and policies in Snort and Bro. Table II shows the attack symptoms detected by Gestalt and their associated belief scores, as well as the alerts triggered by the other two systems. All the experiments were conducted in a controlled LAN. The unpatched operating systems and services were installed, and the packets exchanged were logged by Wireshark [16]. Note that the experiment shown in Table II does not observe any "Flooding" (B_6), because Blaster only does flooding in specific date. However, we still list the belief score with and without "Flooding".

The results show that, the polymorphism does evade the content-signature-based approach; the corresponding rules do not be effective. In Snort, the content-signature-based alerts (C_1, C_2) are not triggered in this case. Since polymorphism only changes the appearance of attack packets and does not affect the attack procedure or behavior, Gestalt's behavior-based approach successfully detects the attack with belief score of 0.6547. The score is raised from 0.0023(B_1), 0.5079(B_2), 0.6049(B_3), 0.6160(B_4) until 0.6547(B_5). The results also demonstrate that adopting stealthy scanning strategies is effective, but not good enough. For slow scans, all the scanning-related alerts (B_1, C_4, and A_1) are still triggered, though far later (in 2.9554, 92.4930, and 110.5 seconds, respectively) than original case (< 0.0001 seconds). Gestalt can still detect the other the four attack symptoms with no delay, as in the original Blaster. For the case of selective scans, all the scanning-related detection rules (B_1, C_4, and A_1) fail. Even so, Gestalt still captures the other four attack symptoms with a final belief score of 0.4914. If "SYN-flooding" (B_6) is observed, than the score is 0.6283. We show that behavior-based approach indeed has the potential to resist the sophisticated evasion techniques.

For the unknown attack, it is evident that content-signature and behavior based techniques cannot detect this unknown attack using existing rules, except the scans. Gestalt successfully detects the "unknown" attack, Sasser, by identifying the attack symptoms (B_3, B_4, and B_5) with a belief score of 0.7124 (or 0.9109 if B_6 is observed). Since attackers may take similar intrusion procedure, Gestalt can take such advantages having potential to detect unknown anomalies. New attack symptoms can also be identified by finding deviated protocol/service behavior. We observed a new attack symptom “Null-TCP-connection”, which is a TCP connection without carrying any application layer message. We guess it is a probe to test if a server is active or not.

In the third experiment, we demonstrate the power of multiple attack symptoms. In Gestalt, intrusion detection is not based on a single symptom of abnormal behavior, but rather on progressive inference and the correlation of a set of attack symptoms. Table III shows the number of occurrences of six attack symptoms found in the trace we collected from a campus core network. It contains more than 45 million flows and 680 million raw packets. Clearly, probing and flooding attack symptoms are quite common in the network; however, the occurrences of the other four symptoms are significantly rare. Although “TCP-portsweep“ and “SYN-flooding” seems to be malicious, they may be caused by normal applications. Therefore, we can check them with the following attack symptoms. Table III also shows the number of consecutive occurrences of two attack symptoms. The observation of consecutive attack symptoms enables the behavior-based approach to achieve a better attack assessment result with few false alarms, compared to only observing individual symptoms. Actually, in this network, there has no attack, so the number of total false positives is only 31+3+3. The false positive rate is quite low considering there are total nearly one million attack symptoms observed.

VI. Conclusion

As attacks continue to evolve, there is an urgent need to develop detection mechanisms that are robust to attack-evasion strategies. To this end, we propose Gestalt, an attack behavior profiling and inference system. It focuses on capturing behavior across multiple levels to reveal more anomalies. Multilevel FSM-based network state tracking and correlation architecture in conjunction with a probabilistic inference model are presented in the paper. Our preliminary experiment results show that the proposed approach can significantly reduce the false positives by considering a sequence of anomalous observations collectively rather than in isolation. And even sophisticated attack variants can be detected. Gestalt is thus a promising alternative to existing content-signature-based and anomaly detection techniques. Future works may include automatically generating protocol FSMs and the attack symptoms, and the notion of normality can be extended to other aspects; not restrict to protocol and service execution. Experiments on other network environment, zero-day attacks and an evaluation of the true positives can be a further study of this work.

Acknowledgment

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.

References

[1] Behavioral Analysis Unit (BAU), United States Federal Bureau of Investigation (FBI), http://www.fbi.gov/hq/isd/cirg/ncavc.htm

[2] P. Li, M. Salour, and X. Su, “A Survey of Internet Worm Detection and Containment,” IEEE Communications Surveys & Tutorials, vol. 10, no. 1, 2008, pp. 20-35.

[3] M. Roesch, “Snort: Lightweight Intrusion Detection for Networks,” Proc. the 13th Conference on Systems Administration (LISA ‘99), 1999, pp. 229-238.

[4] S. Chen, and S. Ranka, “An Internet-Worm Early Warning System,” Proc. IEEE Global Telecommunications Conference (GLOBECOM ’04), 2004, pp. 2261-2265.

[5] C. C. Zou, W. Gong, D. Towsley, and L. Goa, “The Monitoring and Early Detection of Internet Worms,” IEEE/ACM Transaction on Networking (TON), 2005, pp. 961-974.

[6] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, “Network Entropy: Technical report,” CAIDA, Tech. Rep. TR-2004-04, 2004.

[7] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, “A Taxonomy of Computer Worms,“ Proc. the 2003 ACM workshop on Rapid malcode, 2003, pp. 11-18.

[8] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia, “A Behavioral Approach to Worm Detection,” Proc. the 2004 ACM Workshop on Rapid Malcode (WORM ‘04), 2004, pp. 43-53.

[9] X. Jiang, and D. Xu, “Profiling self-propagating worms via behavioral footprinting,“ Proc. the 4th ACM workshop on Recurring malcode, 2006, pp. 17-24.

[10] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier, “Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits,“ ACM SIGCOMM Computer Communication Review, vol. 34, no. 4, 2004, pp. 193-204.

[11] W. Cui, M. Peinado, H. J. Wang, and M. E. Locasto, “ShieldGen: Automatic Data Patch Generation for Unknown Vulnerabilities with Informed Probing,“ Proc. IEEE Symposium on Security and Privacy, 2007, pp. 252-266.

[12] A. Wagner, and B. Plattner, “Entropy Based Worm and Anomaly Detection in Fast IP Networks,” Proc. IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005, pp. 172-177.

[13] G. Vigna, and R. A. Kemmerer, “NetSTAT: A Network-Based Intrusion Detection Approach,” Proc. the 14th Annual Computer Security Applications Conference (ACSAC), 1998, pp. 25-34.

[14] V. Paxson, “Bro: A system for Detecting Network Intruders in Real-Time,” Computer Networks, vol. 31, no. 23-24, pp. 2435-2463, 1998.

[15] S. Staniford, V.Paxson, and N. Weaver, “How to 0wn the Internet in Your Spare Time”, in Proc. 11th USENIX Security Symposium, 2002, pp. 149-167.

[16] Wireshark. Wireshark: the world’s foremost netwrok protocol analyzer. [Online]. Available: http://www.wireshark.org/

[17] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou, “Specification-based Snomaly Detection: A New Approach for Detecting Network Intrusions,” Proc. the 9th ACM conference on Computer and Communications Security (CCS), 2002, pp. 265-274.

[18] D. Brumley, J. Newsome, D. Song, H. Wang, and S. Jha, “Towards Automatic Generation of Vulnerability-Based Signatures.” IEEE Security & Privacy, 2006, pp. 2-16.

[19] M. Christodorescu, S. Jha, and C. Kruegel, “Mining Specifications of Malicious Behavior,” Proc. the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2007, pp. 5-14.

[20] Gordon Lyon, Nmap Network Scanning, Namp Project, 2009.

[21] J. Pearl, “Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,” Morgan Kaufmann Publishers, San Mateo, California, 1988.

[22] J. Postel, “Transmission Control Protocol”, RFC 793, 1981.

[23] N Provos, “A Virtual Honeypot Framework,” in proc. 13th USENIX Security Symposium, San Diego, CA, August 2004.