DTRAB: Combating Against Attacks on Encrypted Protocols Through Traffic-Feature Analysis

Zubair M. Fadlullah, *Student Member, IEEE*, Tarik Taleb, *Member, IEEE*, Athanasios V. Vasilakos, *Member, IEEE*, Mohsen Guizani, *Fellow, IEEE*, and Nei Kato, *Senior Member, IEEE*

***Abstract*—The unbridled growth of the Internet and the network-based applications has contributed to enormous security leaks. Even the cryptographic protocols, which are used to provide secure communication, are often targeted by diverse attacks. Intrusion detection systems (IDSs) are often employed to monitor network traffic and host activities that may lead to unauthorized accesses and attacks against vulnerable services. Most of the conventional misuse-based and anomaly-based IDSs are ineffective against attacks targeted at encrypted protocols since they heavily rely on inspecting the payload contents. To combat against attacks on encrypted protocols, we propose an anomaly-based detection system by using strategically distributed monitoring stubs (MSs). We have categorized various attacks against cryptographic protocols. The MSs, by sniffing the encrypted traffic, extract features for detecting these attacks and construct normal usage behavior profiles. Upon detecting suspicious activities due to the deviations from these normal profiles, the MSs notify the victim servers, which may then take necessary actions. In addition to detecting attacks, the MSs can also trace back the originating network of the attack. We call our unique approach DTRAB since it focuses on both Detection and TRAceBack in the MS level. The effectiveness of the proposed detection and traceback methods are verified through extensive simulations and Internet datasets.**

***Index Terms*—Computer security, encrypted protocol (cryptographic protocol), intrusion detection system (IDS).**

I. Introduction

CRYPTOGRAPHIC protocols rely upon encryption to provide secure communication between involved parties. A wide range of cryptographic protocols are employed by popular applications and services to ensure data confidentiality, integrity, and authentication. For example, Secure Socket Layer (SSL) and its successor Transport Layer Security (TLS) are extensively used to provide authentication and encryption in order to transmit sensitive data. Secure Shell (SSH) has become highly popular for providing password-based authentication and remote logins. Cryptographic protocols have also been developed for the network level [1], such as IPSec, which is extensively used in virtual private networks (VPNs). The purpose of all these encrypted protocols is to resist malicious intrusions and eavesdropping. It is, however, ironic that the network services and applications become vulnerable once the underlying encrypted protocols get compromised.

The number of attacks against encrypted protocols has increased significantly in recent times. With the evolution of high-speed Internet and processing power, it is only natural to assume that more sophisticated attacks will emerge and pose serious threats to encrypted protocols. For instance, adaptive-chosen cipher-text attacks were considered to be only theoretically possible until 1998, when Daniel Bleichenbacher successfully carried out an attack against systems using RSA encryption in concert with the PKCS #1 v1 encoding function [2]. This raised a huge concern among the networking community since the versions of SSL protocol used by thousands of Web servers were then vulnerable to this type of attack. More innovative attacks like the Bleichenbacher attacks took advantage of flaws within the PKCS #1 function to gradually reveal the content of a RSA encrypted message. Although these attacks involved the transmission of several million trial-and-error-based ciphertexts to the encrypted Web servers, they practically implied that a SSL session key could be exposed in a reasonable amount of time, perhaps a day or less. Later on, the timing attack devised by Boneh and Brumley [3] extracted private keys from an OpenSSL-based Web server in less than 6 h, leading to a fascinating breakthrough in the field of network security. Similar attacks against SSH also came into use, such as Portable OpenSSH PAM timing attacks [4], in which an attacker could determine the existence of a given login by comparing the time the remote SSHD-daemon took to refuse an invalid password for a nonexistent login to that for a valid login.

As it is evident that these attacks do exist in practice, it is imperative that these threats be detected as early as possible in order to thwart them. Our topic of interest is a distributed detection mechanism that is able to detect the anomalous events as early as possible, especially before significant damage is inflicted on the victim by the attacker. The coordination of distinct agents monitoring the network flows at different points requires an appropriated architecture that must be developed. We address these issues in our paper effectively and attempt to design adequate solutions to these problems. We propose the DTRAB scheme, which is not limited to constructing a defensive mechanism to discover attacks; we devise an aggressive countermeasure that not only detects a potential threat, but also investigates

Manuscript received October 22, 2008; revised July 24, 2009; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor D. Agrawal. First published January 19, 2010; current version published August 18, 2010. This work was supported in part by Global COE program, Japan, and the European Commission in the context of the Research Framework Program Seven (FP7) project ResumeNet (Grant Agreement No. 224619).

Z. M. Fadlullah and N. Kato are with Graduate School of Information Sciences (GSIS), Tohoku University, Sendai 980-8579, Japan (e-mail: zubair@it.ecei.tohoku.ac.jp; kato@it.ecei.tohoku.ac.jp).

T. Taleb is with NEC Europe Ltd., Heidelberg 69115, Germany (e-mail: tarik.taleb@nw.neclab.eu).

A. V. Vasilakos is with Department of Computer and Telecommunications Engineering, University of Western Macedonia, Kozani 50100, Greece (e-mail: vasilako@ath.forthnet.gr).

M. Guizani is with Kuwait University, Safat 13060, Kuwait (e-mail: mguizani@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2009.2039492

the root of the threat by attempting to trace back the attacker's original network or subnetwork.

The remainder of this paper is organized as follows. Section II surveys some related work on intrusion detection and traceback systems while exploring the shortcomings of these contemporary approaches to deal with cryptographic attacks. Section III at first presents the scope of attacks that may be detectable by DTRAB. In this section, the network topology that may be considered is then provided, which is based upon the monitoring stubs (MSs). The functionality of the MSs in order to detect and trace back the attacks against cryptographic protocols is then illustrated in four modes, namely learning, detection, alert, and traceback phases. The performance of DTRAB is evaluated in Section IV with the aid of simulations. Due to difficulties in obtaining real encrypted traces that are rather sensitive in nature, this section also demonstrates the application of DTRAB for detecting nonencrypted attacks in Internet datasets. Finally, Section V concludes the paper.

II. Related Work

A. Previous Work on Detecting Attacks Against Encrypted Protocols

Intrusion detection has been an active field of research for over two decades [8], and most conventional IDSs operate by inspecting the contents of the networking packets. Once encrypted, the packet contents are garbled and the intrusion detection systems (IDSs) fail to recognize whether the payloads are normal or potentially malicious. As a result, despite substantial research and commercial investments, traditional IDSs still remain ineffective when they encounter encrypted traffic.

Intrusion detection systems can be broadly categorized in two ways, namely signature-based and anomaly-based detection techniques. A signature-based (also known as rule-based or misuse-based) IDS uses previously stored attack descriptions to compare if a portion of the monitored network packets is malicious. The attack descriptions or attack signatures may be simply stored as patterns or may use complex approaches based on state machines or neural networks [9] to map multiple features to abstract the overall attack manifestation. Since a given signature is associated with a known attack abstraction, a signature-based detector can usually assign names (such as Smurf [10] or Ping-of-Death [11]) to attacks with ease. If a similar attack manifestation is found, signature-based IDSs can identify previously unseen attacks, which are equivalent to known patterns. Having said this, the signature-based detection schemes are inherently incapable of detecting truly novel attacks and suffer from high rates of false alarms when attack signatures match both intrusive and nonintrusive patterns. On the other hand, the primary strength of an anomaly-based detection scheme is its ability to recognize novel attacks. At first, statistical models and artificial intelligence (AI) tools such as neural networks are employed to characterize a normal profile. At the advent of a malicious event, an anomaly-based technique senses a significant deviation from the normal profile. The drawbacks of the anomaly-based IDS include higher false alarm rates and the difficulty in classifying or naming attacks.

Recent research efforts have been devoted toward detecting various attacks against encrypted protocols such as SSL/TLS and SSH. For instance, an attacker launching the infamous remote timing attack [3] against an OpenSSL server could extract the private key stored in the server within 6 h. All the attacker had to do was to measure the time the OpenSSL service took to respond to decryption queries on a trial-and-error basis. Canvel *et al.* [12] illustrated password attacks against Internet Message Access Protocol (IMAP) servers using SSL-tunnel with its peer users. This compromised the security of mail transactions even under encrypted sessions. Version 3.0 of SSL was found to be vulnerable against the Version Rollback attack [13], in which an attacker tricked the server to downgrade its version of SSL to 2.0. By doing so, the attacker could then take advantage of the vulnerabilities associated with the lower version of SSL. Additionally, buffer-overflow attacks [14], [15] against openSSL servers led to denial of service (DoS)-like phenomena. Propagation of the Slapper worm [16] is a notable example that posed a serious DoS threat to Apache Web servers that use the mod-ssl library. McClur *et al.* [17] identified encryption to be the biggest inhibitor to the growth of network-based IDSs. By encrypting traffic over SSL, the commercial Web servers practically blind the network IDS sensors from detecting attacks. The only defense against this shortcoming of the network IDS is on-the-fly SSL decryption technology such as SSL Dump [18]. However, such approaches require a copy of the SSL server's private certificates and present an additional security hazard.

Yamada *et al.* [19] illustrate an encrypted traffic analysis to reinforce the detection of encrypted Web intrusion. Their method focused on analyzing the contents of the encrypted traffic by using only data size and timing without having to resort to decryption. Access information in terms of data size and timing for every Web client was extracted from the encrypted Web traffic by reconstructing the TCP sessions and the headers of the encrypted sessions. Based on the low access frequency of malicious activities, the encrypted traffic analysis statistically detected rare events as anomalies and reported the same as suspicious attacks.

An overlay-based architecture called WebSOS [20], comprising access points, beacons, and servlets, has been conceived to enable a Web server to function even under a DoS attack. The end-to-end communication between a client and the server is secured by SSL sessions. When an access-point is attacked, WebSOS chooses another access point so that traffic from legitimate clients can still enter the overlay. On the other hand, if a node is under attack, the overlay topology is modified by computing new paths to other nodes in the overlay.

In order to thwart the Man-in-the-Middle (MITM) [21] attack against SSL and TLS-based client/server communication, "SSL/TLS session-aware user authentication" has introduced a new approach. In this method, a client first authenticates and provides a credential to a legitimate cryptographic server, which then generates a corresponding user authentication code (UAC) and an initial SSL/TLS session. In the event that a MITM attacker steals the UAC, he cannot modify the UAC contents, which are encrypted. To pretend as a valid client, the attacker then requires to send the UAC using his own SSL/TLS session, which is not the same as the server-generated session. This session awareness prevents the retransmission of any intercepted UAC.

ProtoMon [5], an anomaly-based IDS for both cryptographic and application-level protocols, includes the use of lightweight

protocol monitors to detect a deviation from a previously constructed normal behavior profile. ProtoMon functions in three modes, namely Learn, Detect, and Prevent modes. First in the Learn mode, a monitoring stub per server constructs normal usage patterns for the monitored protocols. In the Detect mode, ProtoMon constantly compares the online observations with the acceptable threshold of normal profiles. Once the system detects an anomaly, it switches to the third and final mode, in which the monitor stub slows down the protocol response so that the anomaly may not go beyond the threshold level. The delay is removed when no more anomaly is detected. Despite the unique features introduced in ProtoMon, there are several significant shortcomings. The use of a simple arbitrary threshold to determine the anomaly is inadequate due to the dynamic changes in the network behavior. Integrating protocol monitors with the protocol library will also affect the system once the protocol libraries require to be updated. The delay imposed in the Prevent mode serves as a mere damage control mechanism. Furthermore, a monitoring agent for each server serves the purpose, but perhaps not as efficiently as compared to a distributed set of monitoring stubs exchanging information. The current work (DTRAB) presents a solution to these problems by using a dynamic thresholding scheme to detect anomaly and by distributing unique monitoring agents over the network topology.

B. Previous Work on Tracing Back Attackers Against Encrypted Protocols

In addition to detecting attacks, the issue of tracing back the attackers has remained a challenging problem over the years. Many researchers have focused on integrating traceback with detection schemes [22]. Some traditional traceback techniques [23] use a variation of the Time-Efficient Stream Loss-tolerant Authentication (TESLA) protocol to generate a code, based on the IP addresses of the routing devices, that sequentially handles packets. By employing a map of the IP addresses of all upstream routers, the victim of an attack can efficiently reconstruct the route of a packet up to 32 devices [24]. These works focus on identifying the route within the network packets without increasing the packet size, which has challenged traceback researchers over the recent years.

Other traceback approaches require the routers to generate additional packets for each packet that passes through the routers. The victim host receives both the original packets and these extra packets, which provide identification of the originating routing devices. The obvious disadvantage of this approach is an increase in the network traffic. In order to deal with this, [25] proposes an extra "trace-packet" to be generated on a probabilistic basis, for instance approximately one trace-packet for every 20 000 packets. This approach works for attacks involving a large number of attack packets (e.g., TCP SYN-flood) lasting for a reasonable length of time. However, an attack causing a lower volume of attack packets can evade this system since enough trace-packets are not generated for successful reconstruction of a path back to the attack-host.

Traceback techniques such as probabilistic packet marking (PPM) [26] and its enhanced variant [27], and other packet marking schemes [28] including determinsitic packet marking (DPM), ICMP traceback (iTRace), and logging techniques [28] such as Source path Isolation Engine (SPIE) require the IP header information. This requirement poses difficulty in tracing back an attacker that sends encrypted packets since the traceback modules need to decrypt the headers of the attack packets. However, decrypting packets at intermediate monitoring agents will not be effective since this will contribute to significant overheads and violation of privacy.

One of the most common techniques to evade detection is the use of "stepping stones" [29], where an attacker often masks his identity by launching attacks from intermediary hosts that were previously compromised. This enables the attacker to use a chain of interactive connections using protocols such as SSH to dispatch malicious commands over the "stepping stone" chain to gain access to the victim machine. It is, indeed, difficult to trace back the trail of the attacker owing to the sheer volume as well as the chaotic nature of the traffic on the Internet. The final victim can, at best, see the traffic from the last hop of the chain of the stepping stone. In quest of tracing a stream of attack packets through a number of "stepping stones," content-based stream-matching approaches came into use. One notable example of such an approach is "Thumb-printing" [30], which shows good performance in tracing back stepping-stone attacks involving nonencrypted protocols only. Alternate approaches include correlation methods based on interpacket-delay (IPD) [31], [32] for tracing back attacks against encrypted connections. IPD remains as a distinctive feature in normal interactive connections that employ encrypted protocols such as SSH. By correlating IPD of different connections across the network, this approach identifies whether the inspected packets belong to the same connections. Because there are different correlation points in the experimental setup, connections that are highly correlated can be tracked in a reverse way.

Blum *et al.* [33] proposed an algorithm based on the distinctive characteristics such as packet size and timing information of the interactive traffic rather than the packet contents. Using the algorithm, it was possible to find stepping stones even when the traffic was encrypted. The timing-based algorithm performed more efficiently compared to the traditional context-based techniques. Blum *et al.* investigated not only the detection of interactive stepping stones, but also made attempts to determine an algorithmic bound over the detection approach. The stepping-stone detection problem sheds some light on the difficult ordeal of tracing back attacks against encrypted protocols.

Mansfield *et al.* [34] present a traceback scheme that also relies on correlation-based techniques. Their work is based on the observation that flow patterns for normal network usage are considerably different than those under an attack. According to this theory, RMON devices acting as probes placed at different points of the network monitor the TCP-SYN connections during normal scenario and attack scenarios involving TCP-SYN packet-based DoS attacks. By correlating traffic patterns at various points of the network, the probes look for the presence of similar flows at other probes to establish the attack path. However, this approach does not investigate whether it can be extended to tracing back attacks against encrypted protocols.

III. Proposed Detection and Traceback Schemes—DTRAB

The previous sections revealed that more attention needs to be paid in detecting and tracing back attacks against encrypted protocols, as contemporary techniques fail to adequately combat

with these threats. In this section, we propose DTRAB, which hinges on its ability to detect anomalies in the protocol behavior that serve as indications of attacks.

A. Envisioned Attacks

In this section, we attempt to clarify the different attack classes that DTRAB may address. For instance, the OpenSSL implementation of SSL is particularly vulnerable to specific remote timing, MITM, buffer overflow, and version rollback attacks. In the remote timing attack, SSL renegotiation attack, and password attack (also known as dictionary attack or brute force attack) against SSH, there is a high interaction between the attacker and the cryptographic protocol server. Scanning attacks that examine the existence and configuration of a generic Web server or a proxy server (or even an encrypted HTTPS-enabled server) at an IP address also contribute to a high volume of attack features. Directory-traversal attacks also exhibit similar characteristics. We broadly call these attacks the "highly interactive attacks."

On the contrary, few messages are exchanged between the client and the server in a buffer overflow attack, which can execute arbitrary codes on the victim server by overwriting stack or heap memory of the process. We term these attacks as "low interactive attacks." Cross-scripting languages that lead to attack vulnerabilities may also be categorized as "low interactive attacks." Since a lot of applications (e.g., patches, antivirus, malware detection tools) are used to counter these low interactive buffer-overflow and cross-scripting attacks at the hosts, we mainly provide examples on detecting highly interactive attacks in the rest of the paper.

B. Expected Network Topology

We intend to implement the IDS and traceback entities aside network elements, such as edge or border gateway routers. The obvious reason behind this choice is to avoid the additional computational load and memory overhead occurring at the server if the detection and traceback modules are integrated with the servers. Consequently, this paper contributes by envisioning uniquely designed IDSs, which we call monitoring stubs, or simply MSs. In contrast with the monitoring agents dedicated to each server as proposed in [5], the proposed MSs are distributed (e.g., aside gateways, edge routers, and some of the selected core routers) over the entire network topology.

An example scenario of the envisioned network topology is shown in Fig. 1, which consists of a number of servers running services based on both encrypted and application-level protocols. Users from an untrusted network or from the Internet may connect to any one of these servers. Seven MSs are placed aside the network elements. The MSs, by sniffing, monitor the traffic headers but do not inspect the payloads. When an attack is launched by a host (in Network-1), say from the untrusted network to victim server 1, MS_3, MS_2, and MS_1 consequently observe an influx in abnormal protocol operations interpreted as an attack feature. In the remainder of this section, we shall describe how the MSs effectively detect attacks against encrypted protocols and try to trace back the attacker. Furthermore, by specifying the normal operation modes and request-for-comments (RFCs) specifications of different protocols in the MS' databases, this approach may also be extended to detect attacks

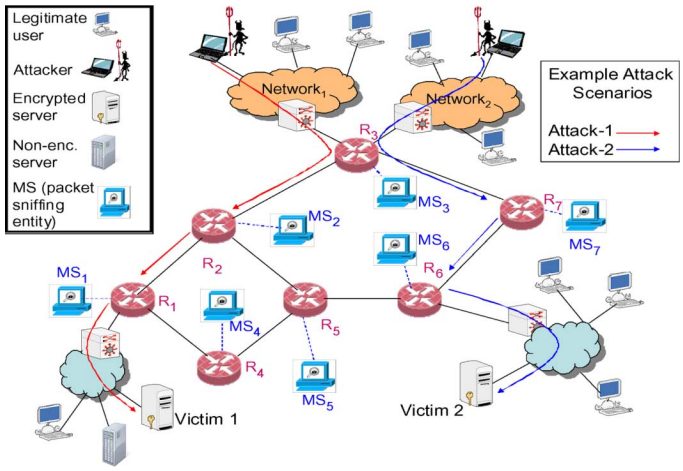


Fig. 1. A sample architecture with added MSs.

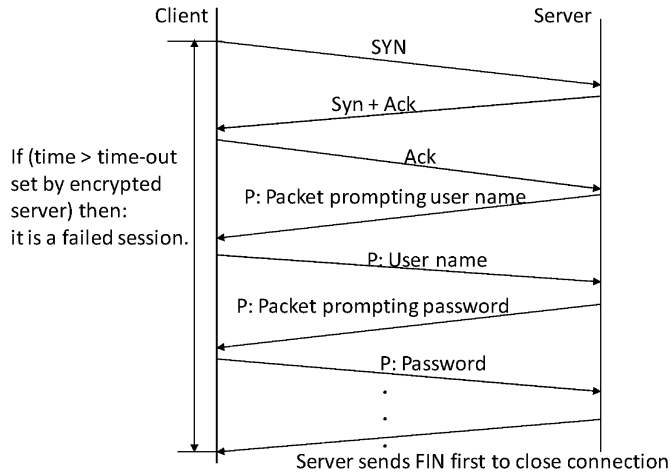


Fig. 2. Connection-flow-based failed session formulation for SSH protocol.

against standard application-level protocols. For ease of understanding, we include two examples demonstrating the deviations from normal operations of two encrypted protocols (SSH and HTTPS, respectively) under attack that may be considered as possible attack features.

Since a MS is only a packet-sniffing entity located aside a router, it does not slow down the network traffic. In case of application level protocols, it is a trivial task to sniff both the network packet headers and the payload contents and to inspect and analyze the information afterward. For encrypted protocols, a MS needs to adopt a different approach. A MS utilizes the TCPDUMP [36] tool to monitor the TCP headers that are not encrypted. For example, in order to detect a failed SSH session due to a password-based attack against SSH-based services on port 22, a MS requires to know how the SSH protocol works in the transport layer level. At first, a client attempts to establish a connection to the server by sending a SYN packet as shown in Fig. 2. The server acknowledges this by sending an ACK and a SYN packet of its own. If the client manages to successfully log onto the server and wants to quit, the client will initiate the FIN packet first. This is a normal mode of operation in SSH. On the contrary, if the server initiates the FIN packet first, it indicates that the server is shutting down the connection because of either an invalid attempt to access the service or a timeout. A

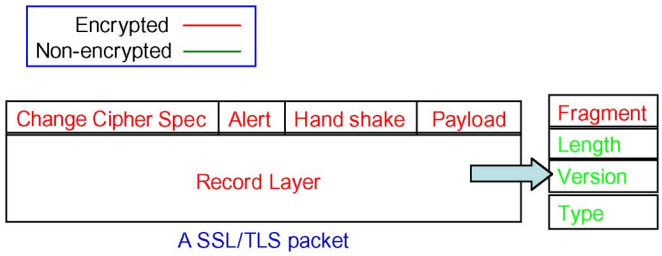


Fig. 3. SSL/TLS packet structure.

MS monitors such connection flows, and when it discovers that the server is the first originator of the FIN packet soon after the connection attempt, it recognizes a deviation in the protocol's normal mode of operation and deems that event as a "failed session."

By analyzing encrypted Web traffic flows using SSL/TLS, a MS may also see abnormal protocol operations in terms of the ratio of the request size to the corresponding response size. In case of "low interactive attacks," which are mostly launched against encrypted Web servers offering HTTPS services, a MS may look to extract such information and consider it as a potential attack feature. With nonencrypted HTTP traffic, this can be readily computed by inspecting the packet contents. However, extracting the request and response sizes from the HTTPS headers in encrypted traffic is indeed difficult since these headers are also encrypted. Furthermore, random paddings of 255 bytes are added to the packets in SSL/TLS.

For this purpose, the MSs sniff packets destined for port 443 and look for client requests. If packets are observed continuously, they are considered to belong to a single activity, such as clicking a URL or requesting a file to download. As the MS determines such an activity, it starts reconstructing the corresponding TCP sessions from the headers. Once the TCP sessions are established, the MS can decode the SSL/TLS session to obtain the sizes of the request and corresponding response from the server. This is possible because every SSL/TLS packet is structured in such a manner (as shown in Fig. 3) that the "Fragment" header of the "Record Layer" is encrypted while the remaining three headers, namely "Length," "Version," and "Type," are not encrypted. The "Length" header provides an estimate of the size of either the request or response packets, depending on the directionality of the connection in the considered flow.

It should be noted that such attack features are not manually selected; rather, each MS maintains a database of RFCs specifications pertaining to the usage of various encrypted and nonencrypted protocols. The functionality of a MS is manifold, including learning normal profiles, monitoring for deviations from normal protocols operations, generating alerts, and finally tracing back the attacker. The operational modes of a MS are described below.

1) DTRAB Learning Phase: To accurately identify anomalous behaviors, it is essential to study the protocol implementations and standard documents such as RFCs, from which we can delineate the normal mode of the protocol operations. Additionally, the protocol behavior in a network is subject to changes in different periods of a day. For instance, a corporate Web site may be accessed heavily during the day and not so much at night. To inflict these factors, a statistical profile over time may be developed in the learning phase during the normal network conditions or near normal network conditions with low acceptable levels of suspicious activities. Each MS in the envisioned topology creates a database with features extracted from the nonencrypted headers of the monitored traffic over time.

For identifying attacks, the MS monitors various protocol operations for anomalous modes of operation (e.g., number of failed sessions) and records them in such a manner so that they may serve as parameters to the nonparametric Cusum algorithm used by the MSs in the detection phase. The format of a typical table from the database is shown in Fig. 4. The two fields, S_n and T_n, which indicate the number of failed sessions and number of total sessions, respectively, are sampled over the profiling interval Δ_n. Using these parameters, the fraction of failed sessions, F_n, is then computed and stored in the database.

2) DTRAB Detection Phase: The detection approach adopted in DTRAB involves detecting anomalies. This relies on detecting the point of change in the encrypted protocol behavior as quickly as possible under an attack. For this purpose, we employ the nonparametric Cusum algorithm, which is a statistical tool. The impact of the statistical application of the nonparametric Cusum algorithm in the analysis of attack features extracted from the packet headers in a unique manner is our contribution. Our ingenuity lies in how we treat the problem of cryptographic attack detection and apply the statistical tool. We realize that it is expensive to employ the classical version of the Cusum algorithm and other change-point detection algorithms due to the manner in which they demand to learn about statistical probabilities of hypotheses of the normal and abnormal events *a priori*. Such hypotheses are referred to as parameters. Furthermore, Internet traffic cannot be modeled appropriately based on such hypotheses. This is why we choose to adopt the nonparametric version of Cusum, which is a lightweight algorithm applicable to the traffic in the Internet, including the scope of encrypted traffic. It is to be noted that the term "nonparametric" implies that the scheme may be adopted without having any knowledge of the traffic distribution beforehand. We employ the nonparametric Cusum algorithm at the MSs to detect points of changes in the network behavior at the advent of an anomaly. The ability of this scheme to detect minute changes in the network profile, and the ease with which it can be deployed at the MS-level encourages us to select the nonparametric Cusum algorithm as the core detection tool.

Here, we give an example of the proposed detection method by analyzing a random sequence of the number of failed SSH sessions in a considered network flow over time. The fraction of failed sessions, F_n, is obtained in a time interval Δ_n. Here, n is the number of monitored profiles, and the profiling periods are of same temporal lengths, i.e., $\Delta_{n+1} = \Delta_n$. Let m be the mean of F_n over the profiling period of normal scenario. For the normal SSH traffic, the value of m is way below 1 and remains close to zero until the system senses an anomaly [Fig. 5(a)]. Hence, F_n may be considered as a stationary stochastic process, and variation of the mean of F_n can be reported by Cusum algorithm.

To account for the negative drift in Cusum algorithm, we assume that the mean value of the random sequence is negative during the normal conditions and becomes positive when a change takes place. In order to conform with this assumption, it

	Number of Failed Sessions, S_n						Total number of Sessions, T_n						Fraction of Failed Sessions, $F_n = S_n/T_n$							$G_n = F_n - \beta$					
Profiling Interval Size, Δ_n	Δ_n=5s			Δ_n=10s			Δ_n=5s			Δ_n=10s			Δ_n=5s			m*	Δ_n=10s			m*	Δ_n=5s			Δ_n=10s		
Protocol	n	time(s)	S_n	n	t(s)	S_n	n	t(s)	T_n	n	t(s)	T_n	n	t(s)	F_n		n	t(s)	F_n		n	t(s)	G_n	n	t(s)	G_n
OpenSSH	1	1-5	2				1	1-5	2290				1	1-5	8.73E-4	0.9 E-4	1	1-10	1.3 E-3	0.9 E-4	1	1-5	-0.0078			
	2	6-10	4	1	1-10	6	2	6-10	2183	1	1-10	4473	2	6-10	1.83E-4						2	6-10	-0.0069	1	1-10	-0.0068
	3	11-15	0				3	11-15	2129				3	11-15	0		2	11-20	4.6 E-4		3	11-15	-0.0087			
	4	16-20	2	2	11-20	2	4	16-20	2205	2	11-20	4334	4	16-20	0.90E-4						4	16-20	-0.0086	2	11-20	-0.0075
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...		...	...	...		...	...	...	...	...	...
OpenSSL	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...		...	...	...		...	...	...	...	...	...

* m = ($\sum F_n$)/(number of observations during n)

Fig. 4. A part of the database format created at a monitoring stub for profiling.

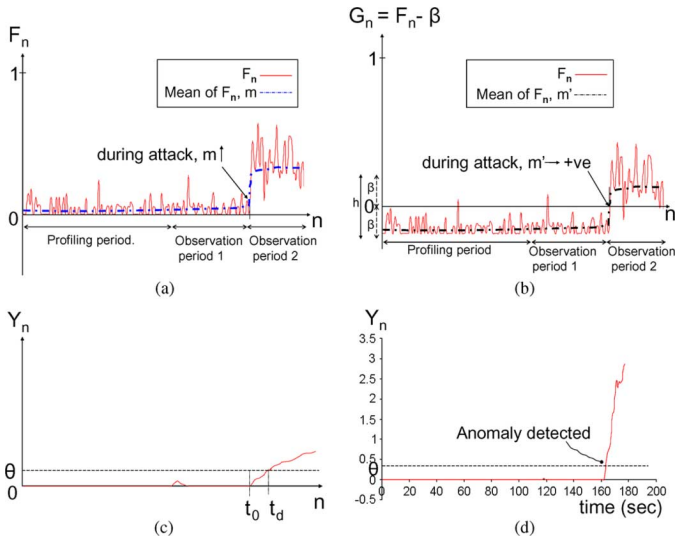


Fig. 5. Computing nonparametric Cusum. (a) F_n and m versus n. (b) G_n and mean of G_n versus n. (c) computing the threshold θ. (d) Y_n versus time: attack is detected when Y_n exceeds θ.

is necessary to transform F_n into a new sequence G_n. The required transformation is obtained by $(G_n = F_n - \beta)$ as shown in Fig. 5(b), where β is the transformation parameter such that $(\beta \gg m)$.

In our scheme, we compute β by taking the average of the highest F_n values obtained during the profiling period. At the advent of an attack, the mean of G_n noted by m', eventually increases considerably in contrast with that during the profiling period. This increase of m' can be lower-bounded by h, which is typically set as twice the value of β. The problem of the online detection of attacks is solved by employing a recursive version of the nonparametric Cusum algorithm defined by a new sequence denoted by Y_n, which is as follows:

$$Y_n = [Y_{n-1} + G_n]^+; \quad Y_0 = 0 \tag{1}$$

where $x^+ = x$ if $x > 0$; otherwise, $x^+ = 0$.

A large value of Y_n indicates the presence of an anomaly, i.e., a potential attack. A constant θ, for this profiling, is then required to be adjusted as the threshold to detect the change point that occurs due to an attack.

Let $d_\theta(.)$ be the decision at time n, which indicates normal operation by retaining the value zero and an abnormal situation by retaining a value of 1.

$$d_\theta(Y_n) = \begin{cases} 0, & \text{if } Y_n \le \theta \\ 1, & \text{if } Y_n > \theta. \end{cases} \tag{2}$$

In other words, $d_\theta(Y_n) = I(Y_n > \theta)$, where $I(.)$ is the indicator function. In order to study the detection time, the following terms are then defined:

$$t_d = \inf\{[n : d_\theta(.) = 1]\} \tag{3}$$

$$\rho = \frac{(t_d - t_0)^+}{\theta} \tag{4}$$

where ρ is the normalized detection time. t_0 and t_d indicate the starting time of the attack and the minimum expected detection time after the occurrence of the change due to the attack, respectively. inf means the greatest lower bound [7]. If h is the actual increase in the mean of G_n during an attack, then ρ can be expressed as

$$\rho \to \frac{1}{h - |\beta - m|} \tag{5}$$

where $(h - |\beta - m|)$ is the mean of G_n when $n > t_0$ (i.e., after an attack starts). h serves as an upper bound of the actual detection time. Thus, (5) provides a rough estimation of the actual detection time.

From (4) and (5), θ is computed as expressed by the following equation as shown in Fig. 5(c):

$$\theta \simeq [h - (\beta - m)] \cdot (t_d - t_0)^+ = (\beta + m) \cdot (t_d - t_0)^+ \tag{6}$$

We now attempt to provide an algorithmic bound for G_n in a scenario where two attackers simultaneously introduce the same number of failed sessions over time T. Each attacker then obtains the same cumulative score G_n that captures the behavior of all failed session samples over T. In order to derive an expression for G_n, we define a cutoff point κ such that if $t_{\text{sub}} < \kappa$, it is more inclined to be a malicious sample where t_{sub} denotes the time between two subsequent failed sessions triggered within the considered attack flow. Since we set $G_0 = 0$, we have $G_t < 0$ for all $t > 0$, and hence the CUSUM threshold θ denotes the minimum number of failed session samples required to decide if a remote host is malicious. As a result, we can replace G_n of (1) with $(-t_{\text{sub}}/\kappa)$. Thus, in DTRAB, we incorporate our extended parameters to the nonparametric Cusum approach.

In the absence of attacks, the G_n values lie mostly lower-bounded by $(-\beta)$. During an attack, the G_n values become positive and substantially large. It was mentioned earlier that in our approach h is set as roughly twice the value of β. This is set as a tolerable margin of the change of the mean of G_n due to the anomaly. $(t_d - t_0)$ is usually set to a small value (e.g., 1 s) for quickly detecting an anomaly.

At the detection phase, the MS computes Y_n over time. The Y_n sequence will remain close to zero, i.e., along the horizontal time axis as long as normal conditions prevail in the network. It should be noted here that the MSs in the proposed DTRAB scheme monitor only the initial handshakes in the flows that establish the cryptographic sessions and extract the necessary attack features (e.g., failed sessions). Therefore, the MSs do not maintain all state information pertaining to every connection in individual flows, which would have raised scalability issues.

3) DTRAB Alert Phase: When the nonparametric Cusum algorithm detects an anomaly, the Y_n sequence begins to increase. Once Y_n exceeds θ, the MS generates an alert [Fig. 5(d)] to the server and the neighboring MSs. The MS can also request the server to slow down the protocol response in an attempt to thwart intrusions such as remote timing attacks.

Finally, the MS switches to the trace back mode to identify the attacker, which will be described in the next subsection.

4) DTRAB Traceback Phase: The proposed traceback mechanism relies on the collaboration of the MSs to correlate monitored abnormal operations of the protocols (e.g., failed session rates) over time. Every MS, in its database, stores the information of failed sessions that it observes for both incoming and outgoing traffic. This database also contains a list of collaborating MSs, with which the MS can contact in order to reconstruct the attack path. Once an attack is detected, the MS closest to the victim encrypted server goes to the "Trace-back mode." An example of how the proposed traceback scheme functions is illustrated in Fig. 6. Here, R is the considered confluence point [41], where traffic from disparate sources converge. When MS_a requests MS_b to compare the monitored failed sessions in the direction as shown, MS_b starts to correlate the features of its outbound traffic $(S_{\mathrm{out,MS}_b})$ with that of each of the incoming flows into the confluence point R, i.e., $[d_{\mathrm{in},1}, d_{\mathrm{in},2}, \ldots, d_{\mathrm{in},n}]$ where $d_{\mathrm{in},n}$ refers to the features of the nth incoming flow at R. A strong correlation coefficient indicates a possible path back to the origin of the attack. MS_b can now collaborate with MSs along that path to continue traceback even further.

Every MS monitors the failed sessions in time-slots, λ, and in a window W, which consists of an integral number of λs. Thus, the pattern is defined by the length of the time-slot (λ), the size of the "window" (W), and the monitored features in each slot in the window. We use these metrics to define the vector V

$$V = \lambda, F_1, F_2, F_3, \ldots, F_l, \ldots, F_W \tag{7}$$

where F_l is the failed session rate monitored during the lth time slot.

Let the one-way propagation delay between the aforementioned pair of MSs (MS_a and MS_b) be ξ. At MS_a, the monitoring commences at time t and continues till $[t + (W * \lambda)]$, as depicted in Fig. 6. Thus, the vector, V_a can be constructed at MS_a by

$$V_a = \lambda, F_{a_1}, F_{a_2}, F_{a_3}, \ldots, F_{a_l}, \ldots, F_{a_W}. \tag{8}$$

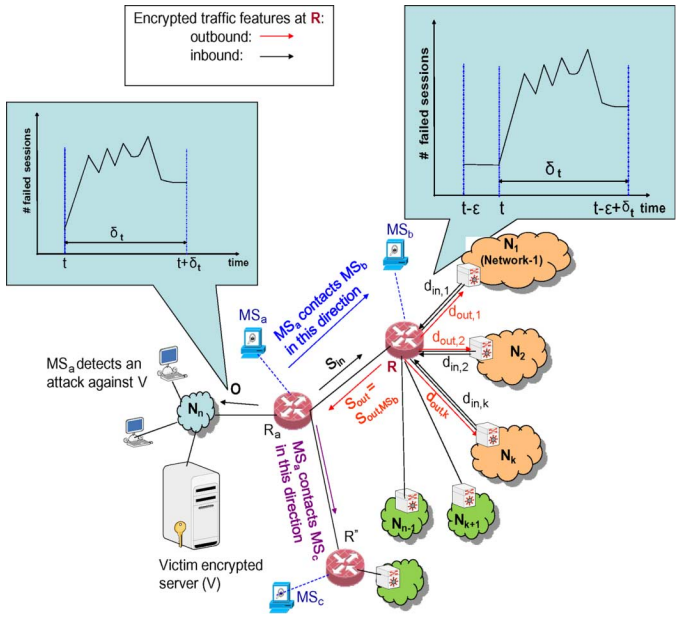


Fig. 6. MS_b compares the outbound traffic feature (S_{out}) to the features of each inbound flow at the confluence point R.

As shown in Fig. 6, at MS_b, the monitoring of F_l begins at time $(t - \xi)$ and goes on till $[(t - \xi) + (W * \lambda)]$. Thus, V_b is constructed at MS_b as follows:

$$V_b = \lambda, F_{b_1}, F_{b_2}, F_{b_3}, \ldots, F_{b_l}, \ldots, F_{b_W}. \tag{9}$$

The correlation coefficient, denoted by $c_{a,b}(V_a, V_b)$ (referred to as $c(V_a, V_b)$ in short), between the target vectors, V_a and V_b, is obtained by [34]

$$c(V_a, V_b) = \frac{1}{W \cdot \sigma_a \cdot \sigma_b} \sum_{l=1}^{N} \left[V_a(l) - \dot{V_a}\right] \left[V_b(l) - \dot{V_b}\right] \tag{10}$$

where $\dot{V_a}$ and σ_a stand for the mean and standard deviation of the monitored features of V_a, respectively.

$$\dot{V_a} = \frac{\sum_{l=1}^{N} F_{a_l}}{N} \tag{11}$$

$$\sigma_a^2 = \frac{\sum_{l=1}^{N} \left(F_{a_l} - \dot{V_a}\right)^2}{N}. \tag{12}$$

The value of $c(V_a, V_b)$ lies between $\{-1, 1\}$. If $c(V_a, V_b)$ equals 1, it suggests that the two patterns, represented by the vectors V_a and V_b, are perfectly matched. If the correlation coefficient value is close to 1 (e.g., 0.8 or above), the vectors V_a and V_b are said to be strongly correlated. If $c(V_a, V_b)$ yields a value of zero or close to zero, the compared vectors are not correlated. A negative value of $c(V_a, V_b)$, on the other hand, implies that V_a and V_b are completely opposite of each other.

By applying this to the scenario in Fig. 6, an example set τ with k number of strong correlations between the failed sessions of the inbound flows and the outgoing flow (S_{out}) observed by MS_b may be found as follows:

$$\tau = [c(d_{\mathrm{in},1}, S_{\mathrm{out}}), c(d_{\mathrm{in},2}, S_{\mathrm{out}}), \ldots, c(d_{\mathrm{in},k}, S_{\mathrm{out}})]. \tag{13}$$

Now, instead of a single attack, if there are distributed DoS (DDoS)-like multiple attack sources, the traceback operation may not be sufficient due to convergence of disparate traffic at the confluence R. To address this issue, we formulate the following model. Let us assume that the scenario in Fig. 6 consists of attack traffic from hosts belonging to networks 1 up to k. On the other hand, networks $(k+1)$ up to $(n-1)$ do not participate in the attack. Then, the malicious and legitimate parts of the overall input traffic, denoted by M_{in} and L_{in} may be represented as follows:

$$M_{\text{in}} = \sum_{i=1}^{k} d_{\text{in},i}, \quad L_{\text{in}} = \sum_{i=k+1}^{n} d_{\text{in},i}. \tag{14}$$

Also, let the outbound traffic for the victim network N be represented by O and the rest of the outbound traffic be denoted by $L_{\text{out}} = \sum_{i=1}^{n-1} d_{\text{out},i} = \sum_{i=1}^{n}(1-c_i)d_{\text{in},i}$, where $c_i = c(d_{\text{in},i}, S_{\text{out}})$. Since the total input traffic into the backbone network and the total oubtound traffic from it should be equal (i.e., $M_{\text{in}} + L_{\text{in}} = O + L_{\text{out}}$), we may write

$$\sum_{i=1}^{k} d_{\text{in},i} + \sum_{i=k+1}^{n} d_{\text{in},i} = O + \sum_{i=1}^{n}(1-c_i)d_{\text{in},i} \tag{15}$$

From (15), we can derive O as follows:

$$\begin{aligned} O &= \sum_{i=1}^{n} c_i d_{\text{in},i} = \sum_{i=1}^{k} c_i d_{\text{in},i} + \sum_{i=k+1}^{n} c_i d_{\text{in},i} \\ &= \sum_{i=1}^{k} c_i d_{\text{in},i} + x; \quad (0 \le c_i \le 1). \end{aligned} \tag{16}$$

Then, the problem of the traceback problem at a confluence point, where some traffic flows may contribute to attack features while some may not, can be represented as an objective function as follows:

$$\begin{aligned} &\text{Minimize } |x|^2 = x_1^2 + x_2^2 + x_3^2 + \ldots + x_m^2 \\ &\text{subject to } O = \sum_{i=1}^{k} c_i d_{\text{in},i} + x; \quad (0 \le c_i \le 1). \end{aligned} \tag{17}$$

The objective function in (17) is a quadratic programming problem. It can be solved for a near-optimal or general solution by applying active set techniques. By minimizing the contribution of the nonattack traffic, the correlated attack flows can be more accurately identified through such a solution.

IV. Performance Evaluation

First, we evaluate the performance of DTRAB with small-scale computer network-based experiments and simulations. The simulations were conducted five times, and the average values are used as results. To verify the performance of DTRAB in the large-scale networks, we also apply the DTRAB detection scheme on different unencrypted Internet traces obtained from CAIDA datasets [40].

A. Performance of the Detection Scheme

1) Highly Interactive Attack Detection:

a) Experimental setup: Since each MS deployed in the considered architecture sniffs and monitors encrypted traffic only, this mechanism is inherently same for all the MSs, and therefore, we do not need to redundantly provide detection performance for every single MS. We considered MSs one hop away from the victim cryptographic server for evaluating the proposed detection mtehod. To evaluate the attack detection by an individual MS, a SSH server was targeted, which ran on OpenSUSE Linux 10.1 with 3.2 GHz processor speed and 1 GB memory. A customized SSH traffic generator was designed at the client end, which ran on Windows XP, and the protocol version used was SSH-2. In normal scenario, the SSH connection arrivals follow a Gamma distribution, shape and rate parameters of which are set to 0.2784 and 0.2260, respectively [38]. The sniffer running on a virtual machine configured on the server acted as the monitoring stub, MS_a in this experiment.

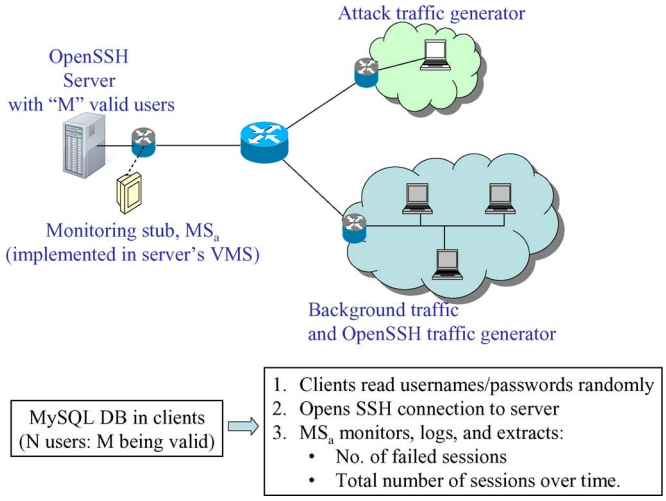


Fig. 7. Experimental setup featuring a MS.

As depicted in Fig. 7, M indicates the number of valid users in the SSH server, while N represents the number of users in the database including both valid and illegitimate users in the client end. Here, $(\mu = (N-M)/N)$ denotes the "attack aggressiveness." Since some users may make typing mistakes in entering authentication information, attack-like features will be injected in the network traffic. However, this is likely to remain considerably low during normal profiling, and we assume $\mu \le 0.05$ to be in the acceptable range to account for the accidentally failed login attempts.

b) Results and analysis:

i) Selection of parameters: At first, a normal usage profile was created by using $\mu = 0.05$ with various sampling intervals $(\Delta_n = \Delta)$ ranging from 1 to 20 s. The F_n sequence behaves as a static stochastic process as its mean during the normal situation remains pretty much the same (close to zero), regardless of the sampling interval sizes. For instance, with $\Delta = 5$ s, m was found to be 0.036 during the profiling period. The transformation parameter β was computed to be 0.31 by taking the average of the upper values of F_n. On the other hand, the threshold θ was 0.34, calculated using (6). To justify our selection of β, we have listed T_{FP}, the time to encounter the first false positive during an observation period of one hour for arbitrarily chosen values of β in Table I. It took a short time to encounter the first false positive for the lower β and corresponding θ values. For higher β values from 0.30 to 0.32, no false positive appeared during

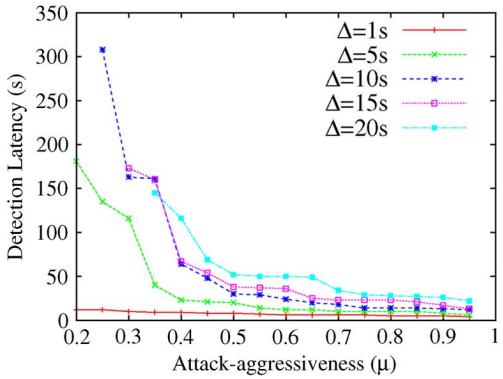


Fig. 8. Detection delays for different attack aggressiveness over various values of Δ.

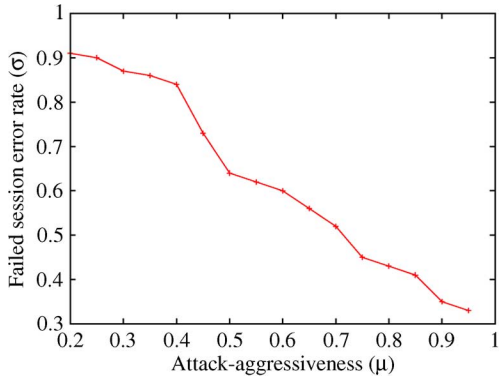


Fig. 9. Failed session error rates (σ) for different values of μ.

TABLE I
TIME TO ENCOUNTER FIRST FALSE POSITIVE (T_{FP}) FOR DIFFERENT VALUES OF β (OBSERVATION TIME: 1 H)

β	T_{FP}(s)		β	T_{FP}(s)
0.05	10		0.22	380
0.08	55		0.25	454
0.1	60		0.28	1,667
0.13	66		0.30	_
0.16	120		0.31	_
0.19	135		0.32	_

the observation period. This conforms with our selection of β by taking the average of the upper values of F_n.

ii) Detection Accuracy: In order to evaluate the detection sensitivity of DTRAB, a number of SSH password attacks with varying attack aggressiveness (i.e., with different values of μ) were then launched against the server. For each μ value, the latency to detect the attack was computed. By varying μ from 0.2 for 0.95 for different values of Δ, the corresponding detection latencies are plotted in Fig. 8. Δ values of 1, 5, 10, 15, and 20 s have been considered in our experiments. Five simulation runs per μ for each case were performed, and the average values of the detection delays are plotted.

The detection latencies drop substantially for increasing μ values for all Δ values in these experiments. For Δ value of 1 s, the highest detection latency is 12 s for the attack with the smallest attack aggressiveness (0.2). It took 4 s, on the other hand, to detect the most aggressive attack $(\mu = 0.95)$. The intermediate attacks with μ values between 0.8 and 0.5 have taken 5 to 8 s.

In the case of $\Delta = 5$ s, the attacks with lower μ took more time to be detected. For instance, the detection latencies for $\mu = 0.2$, 0.25, and 0.3 were indeed high—180, 135, and 116 s, respectively. The reason behind this is the fact that the attack profiles with lower μ do not deviate by much from the normal profile. For μ-values between 0.3 and 0.4, the latencies began to decrease substantially. The average detection delay was approximately 20 s between $\mu = 0.4$ and 0.5. From $\mu = 0.55$, the detection latency started to drop even more and was about 15 s up to $\mu = 0.65$. The average latency for μ between 0.7 and 0.85 was about 10 s. It took the least time, 8 and 6 s, respectively, to detect the most aggressive attacks with $\mu = 0.9$ and 0.95. These results indicate that attacks with aggressiveness below 0.4 are harder to detect, while those with μ over 0.5 are more easily and quickly detectable.

On the other hand, in cases of Δ values of 10, 15, and 20 s, the detection latencies are much more higher. Furthermore, the attacks with low μ are not detected when Δ was set to higher values. This suggests that the sampling time should be as small as possible in order to detect all the attacks with moderately low detection delays. However, $\Delta = 5$ s is more of an ideal choice for these experiments due to the manner the failed sessions are observed by monitoring the TCP headers (Fig. 2). $\Delta = 5$ s is a reasonable amount of time inside which the unsuccessful handshake of SSH or SSL may completely take place.

Now, we introduce a metric called the "Failed session detection error rate," σ, which is expressed by

$$\sigma = \frac{\text{Number of undetected failed sessions}}{\text{Total number of failed sessions}}. \tag{18}$$

Fig. 9 shows σ values for varying values of μ over $\Delta = 5$ s. The results indicate significantly high values of σ for the attacks with lower attack aggressiveness. The reason behind this is the overwhelmingly high number of undetected failed sessions accumulated during the high detection latency in contrast with the moderate number of detected failed sessions during the last sampling interval. For example, the values of σ were 0.91, 0.90, and 0.87, respectively, for $\mu = 0.2$, 0.25, and 0.3. The values of σ decreased gradually along with the aggressiveness of the attack. For attacks with $\mu = 0.45$ and above, the values of σ dropped substantially. The lowest σ (0.33) was encountered during the attack with $\mu = 0.95$. In summary, these results show that the proposed scheme exhibits reasonably small detection delays. In cases of attacks with significantly high attack aggressiveness, the detection delays were only about two sampling intervals. The "Failed session detection error rate" increases when the system is detecting an attack with low attack aggressiveness. Apart from this limitation, the proposed scheme achieves effective detection.

iii) Dynamic Update of θ using multiple Cusum instances: Let us consider two attack scenarios as illustrated in Fig. 10. At first, an attack with $\mu = 0.65$ is initiated, and just before the attack ends, another attack (in the same flow) with lower attack aggressiveness $(\mu = 0.4)$ is launched. With just one instance of the Cusum algorithm, the attacker can evade the detection system.

As soon as the first attack is detected by MS_a, we can reset the Y_n back to zero. However, in case that this attack persists,

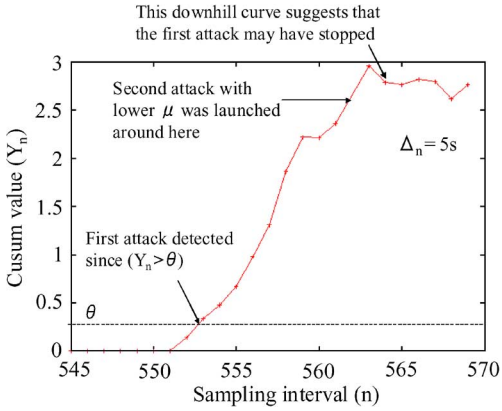


Fig. 10. Detecting two attacks with a single Cusum instance.

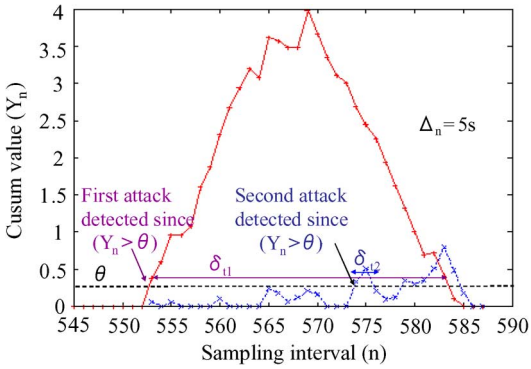


Fig. 11. Detecting two attacks with multiple Cusum instances.

the Y_n value will keep increasing, and in the next interval it will exceed the threshold θ. This will produce more alerts for the same attack that had been detected.

To solve this issue, we use a cluster of Cusum instances (Fig. 11). As soon as an anomaly is detected, the MS creates a new profile excluding the suspicious clients. The MS continues to monitor the Cusum distribution based on the initial profiling to see if the attack subsides. Meanwhile, the second monitoring process continues to run using the most recently created profile. Upon detection of an anomaly in the second instance of the Cusum distribution, the MS will build yet another profile. Furthermore, we can have an estimate of the traceback parameter δ_t in terms of how long to correlate the failed session rates observed by collaborating monitoring stubs.

The nonparametric Cusum algorithm consumed low memory in this experiment. For concurrent attacks, each instance of the detection module took about 5588 to 5592 kB of memory to execute the code. This was roughly 0.3% of the memory available at the MS. The CPU utilization was around 50% with four instances of the detection algorithm running simultaneously.

2) Attack Detection in Internet Data Traces: Due to difficulty in obtaining real network traces for encrypted Web access, we apply our approach to unencrypted access in the CAIDA datasets [40]. The datasets consist of raw traffic traces from the years 2001 to 2009.

In the first analysis, the CAIDA Backscatter-2008 dataset is used, which comprises quarterly week-long collections of responses to spoofed traffic. In this dataset, a DoS-attack victim receives attack traffic with spoofed source IP addresses. Since it cannot differentiate between this spoofed traffic and legitimate requests, it sends responses (i.e., SYN-ACK message of the second step of TCP handshake) to the spoofed sources, which do not reply and complete the TCP handshake by sending an ACK message. DTRAB's detection algorithm learns the fraction of such incomplete sessions from the dataset that comprises predominantly normal traffic and a mixture of some attacks. The monitoring interval Δ_n is set to 5 s without any specific purpose in mind. m, β, and θ are found to be 0.007, 0.009, and 0.016, respectively. Then, by applying DTRAB's detection algorithm in various network domains in the backscatter dataset, anomalous increases in the fraction of the incomplete sessions are detected. We provide two results here by assuming that there are two MSs in two domains where the respective victims are attacked by DDoS with relatively low and high attack rates, respectively. Fig. 12(a) demonstrates the detection of the low-rate DDoS attack during the fourth time interval (i.e., in approximately 20 s) since the beginning of the attack as the cusum value Y_n exceeds θ. In contrast, it takes half the time to detect the second attack, as demonstrated in Fig. 12(b). These observations are indeed similar to the ones found in our simulated experiments with different attack aggressiveness.

We also apply DTRAB's detection algorithm on the code red II worm (2004) and the witty worm (2001) datasets obtained from CAIDA. In this analysis, the MSs in the target networks run Cusum algorithm to look for anomalous protocols behavior. The monitoring interval Δ_n is set to 5 s. One particular anomaly is seen in terms of the degree of outbound connections a given host wants to initiate in a given period of time. Fig. 13(a) demonstrates the DTRAB detection of the code red spread event. The Cusum sequence (Y_n) starts to exceed the threshold $\theta = 150$ after 10 845 monitoring intervals. The reason behind this high detection latency is due to the fact that the worm remains dormant up to this point. CAIDA filtered out most of the sensitive information from the tracefiles, and therefore, other anomalies in the protocol behavior that would indicate the buffer overflow threat posed by the worm could not be analyzed by DTRAB. Nevertheless, since DTRAB detects the anomalous increase in the degree of outbound connections in the network when the worm becomes active on the host, further infections to other hosts could have been prevented by alerting the corresponding server/network admin. In a similar way, Fig. 13(b) shows that the proposed approach can detect the witty worm spread in the 548th interval, which means the dormant period for the witty worm was much less than that of the code red worm.

B. Performance of the Traceback Scheme

Two simulation scenarios are envisioned to evaluate the performance of the proposed traceback scheme. In the first scenario, concurrent attacks are launched against both the victim servers 1 and 2 from $\mathrm{Network}_1$ and $\mathrm{Network}_2$, respectively (Fig. 1). For this scenario, simulations were conducted five times, and the average values are used as results. In the second scenario, the influence of the confluence points on DTRAB's traceback method is investigated and compared to the conventional method [34].

1) Scenario 1: Tracing Back Multiple Attackers:

a) Simulation setup: In this simulation, we investigate the performance of the proposed approach when there are simultaneous attacks from different sources to more than one victim.

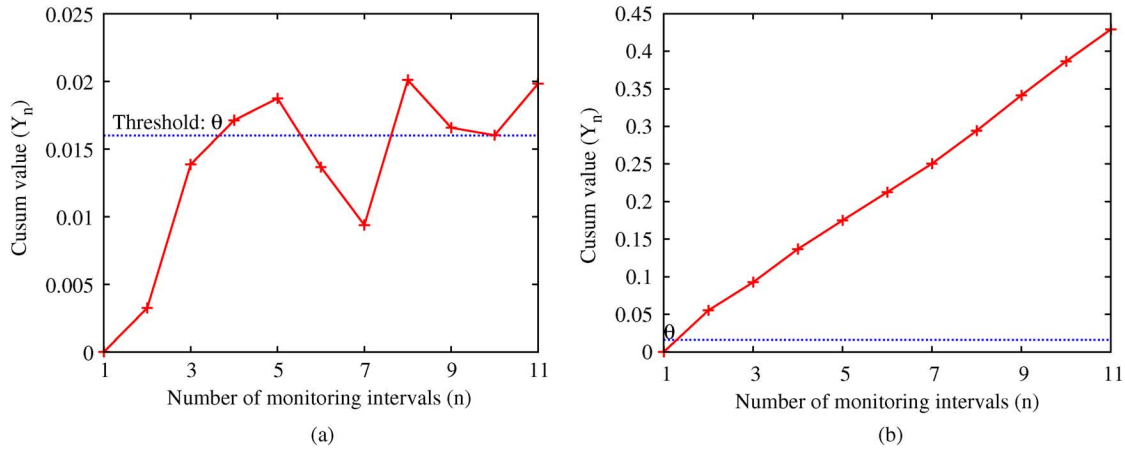


Fig. 12. DTRAB detection of DoS attacks against the victims in two domains. (a) DTRAB detection of a DoS attack with low attack rate. (b) DTRAB detection of a DoS attack with high attack rate.

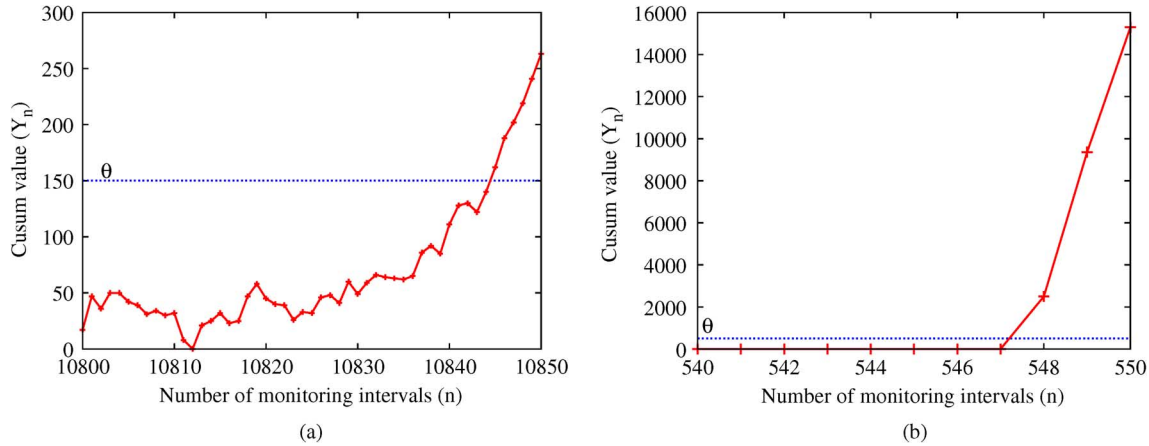


Fig. 13. DTRAB detection of different worm spread events in CAIDA datasets. (a) DTRAB detection of the spread of code red worm. (b) DTRAB detection of the spread of witty worm.

TABLE II
SIMULATION PARAMETERS FOR TRACEBACK SCENARIO 1

SimulationParameters	*Value*
Number of monitoring stubs	7
Dummy Encrypted Protocol	SSH (over TCP)
Background Traffic	CBR, FTP, Telnet
Simulation Time for Trace back	100s
Time slot, λ	1s
Monitoring Window, W	10
No. of times N was monitored	10

By using Network Simulator (NS-2) [39], the topology consisting of seven MSs as shown in Fig. 1 is constructed. The scalability of DTRAB can be demonstrated through the manner in which MSs cooperate with one another to perform traceback. It is worth noting that even in case of fewer number of MSs, since the MSs work by collaborating with other MSs in their neighborhood lists, they would eventually correlate the traffic with far out MSs and still determine the path. The simulation parameters are provided in Table II. SSH password attacks (with varying attack aggressiveness from 0.2 to 0.95) originating from two malicious users in $\mathrm{Network}_1$ and $\mathrm{Network}_2$ are simulated against Victim cryptographic servers 1 and 2, respectively.

b) Results and analysis: Let T_{R_i,R_j} and T_{N_i,R_j} denote the features of traffic directed from router R_i to router R_j and from $\mathrm{Network}_i$ to router R_j, respectively. Following the detection of an attack against victim server 1 (with attack aggressiveness of 0.45), MS_1 starts correlating $S_{\mathrm{out},\mathrm{MS}_1}$ with each of the incoming flows: T_{R_2,R_1} and T_{R_4,R_1}. The corresponding correlation coefficients are 0.924 and (−0.015), respectively. As a result, MS_1 eliminates MS_4 as a subsequent MS to carry on traceback. In the next step, MS_2 is contacted by MS_1. At MS_2, $S_{\mathrm{out},\mathrm{MS}_2}$ is compared to both T_{R_3,R_2} and T_{R_5,R_2}, resulting in the correlation coefficients of 0.934 and 0.0193, respectively. Consequently, MS_2 contacts MS_3, which compares $S_{\mathrm{out},\mathrm{MS}_3}$ to each of $[T_{N_1,R_3}, T_{N_2,R_3}, T_{R_7,R_3}]$. The corresponding correlation coefficients are found to be 0.956, 0.0210, and 0.0303, respectively. This result leads to $\mathrm{Network}_1$, which is indeed the origin of the attack to the victim server 1.

On the other hand, when MS_6 detects the attack against victim server 2, it compares $S_{\mathrm{out},\mathrm{MS}_6}$ with the only incoming flow, T_{R_5,R_6}. The resultant correlation coefficient, 0.0145 is too low to consider MS_5 as the next MS for continuing the traceback. Meanwhile, MS_3 also correlates $S_{\mathrm{out},\mathrm{MS}_6}$ with T_{R_7,R_6}, which yields a strong correlation coefficient of (0.924). This puts MS_7 in charge of the traceback. MS_7 correlates $S_{\mathrm{out},\mathrm{MS}_7}$ with T_{R_3,R_7}, leading to a strong correlation coefficient (0.935).

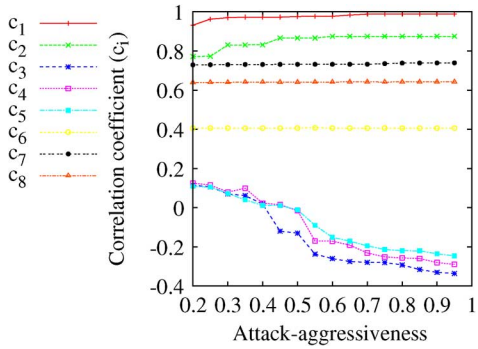


Fig. 14. Comparison of correlation coefficients for different attack aggressiveness.

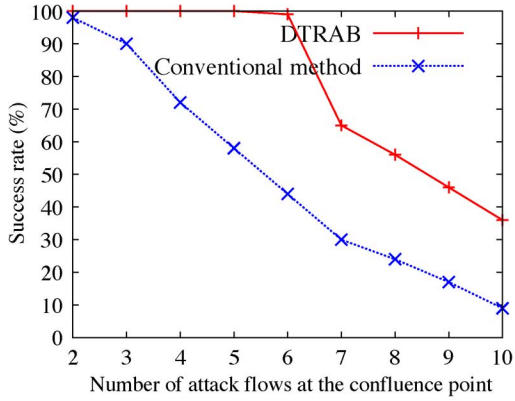


Fig. 15. Success rates of tracing back attacks for different confluence points in DTRAB compared to conventional method.

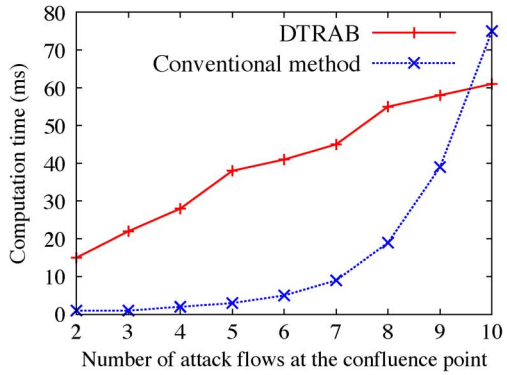


Fig. 16. A comparison of the computation time taken by DTRAB and the conventional approach.

Therefore, MS_7 contacts with MS_3, prompting MS_3 to compare $S_{\mathrm{out},\mathrm{MS}_3}$ with each of $[T_{N_1,R_3}, T_{N_2,R_3}$, and $T_{R_2,R_3}]$. The resultant correlation coefficients are 0.0161, 0.971, and 0.024, respectively. Thus, $\mathrm{Network}_2$, from which the attack was initiated against victim server 2, is discovered.

It may also be interesting to investigate the correlation coefficients for different pairs of flows. In this simulation, we can identify three such pairings: *malicious versus malicious*, *malicious versus normal*, and *normal versus normal* flows. For a single attack against victim server 1 alone (no concurrent attack was launched against victim server 2), let c_1 (between MS_1 and MS_2) and c_2 (between MS_2 and MS_3) represent the correlation coefficients of the vectors both having malicious features. c_3 (in case of MS_1 and MS_4), c_4 (between MS_2 and MS_5), and c_5 (between MS_3 and MS_7) are the correlation coefficients of the vectors, corresponding flows of which have normal features in one and attack features in the other. Lastly, c_6 (for MS_4 and MS_5), c_7 (between MS_5 and MS_6), and c_8 (between MS_6 and MS_7) are the correlation coefficients of the vectors, corresponding flows of which are both normal. Fig. 14 plots the c_1 up to c_8 for various attack aggressiveness ranging from 0.2 to 0.95. Both c_1 and c_2 indicate high correlation values, in the vicinity of one, for various attack aggressiveness. On the other hand, c_3, c_4, and c_5 have correlation values close to zero for different attack aggressiveness. For the higher values of attack aggressiveness, c_3, c_4, and c_5 become negative, indicating further differences between the monitored patterns at the corresponding MS-pairs. These results conform with our traceback decision along the actual attack path. Interestingly though, c_6, c_7, and c_8 have high correlation values and remain more or less the same with varying attack aggressiveness. This is due to the fact that attack traffic did not traverse through the corresponding MSs. However, this does not affect our traceback scheme because the monitoring stubs decide to hand over the charge of traceback operation to the subsequent MSs in such a manner that these comparisons are not encountered.

2) Scenario 2: Investigating the Influence of Confluence Points: In this scenario, we aim at verifying the applicability of DTRAB under the influence of confluence points [41], where large volumes of attack traffic from disparate sources along with normal traffic converge. At a confluence point, when a DDoS attack occurs from multiple sources, the conventional method [34] based on only comparing correlated traffic patterns at different points on the network is not adequate. The DTRAB traceback mechanism addresses this issue by adopting the quadratic model whereby the combination of the DDoS input traffic is also taken into account. For this purpose, real Internet traffic traces collected at the Tohoku University Aobayama campus, Japan, are used and the quadratic problem based model is adopted as shown in (17). To highlight our interest about the confluence point, where both normal and malicious traffic merge, we again refer to Fig. 6. Considering that $\mathrm{Network}_n$ is the victim network, out of the $(n-1)$ inbound flows, k flows contribute to the distributed attack against a host in $\mathrm{Network}_n$. For different values of k from 2 to 10, the success rates of identifying the attack traces using DTRAB are plotted in Fig. 15 and compared to the conventional method based on correlation coefficients alone in [34]. As evident from this figure, in contrast with the conventional approach, DTRAB is able to differentiate the individual attack flows with 100% success rates while the value of k remains below 7. When the number of attack flows at the confluence point increases even more, it affects DTRAB also. However, DTRAB still identifies 40% attack paths back to the actual attack hosts, compared to a meager 9% by the conventional approach. The better performance of DTRAB can be attributed to the fact that, unlike the conventional traceback approach, it also monitors and compares the nonattack flows, which contribute to minimal attack features. However, DTRAB achieves this better performance at the cost of higher computation time, as depicted in the graph in Fig. 16. This graph delineates that the computation time for DTRAB's traceback decision increases almost linearly with the increasing number of attack flows at the confluence. For instance, for two attack

flows in the confluence point, DTRAB takes about 15 ms to identify the actual attack paths back, whereas it takes almost 60 ms to discover 10 attack flows. In particular, it should also be noted that when the number of attack flows at the confluence exceeds six, the computation time of the conventional approach also increases significantly. Therefore, this tradeoff between DTRAB's success rate and computation time is still acceptable.

V. Conclusion

In this paper, we addressed the online detection of attacks against application-level protocols, which are encapsulated inside encrypted sessions. Experiments carried out in the real data network have provided evidence that implementation of the proposed DTRAB in the monitoring stub (MS) is feasible. DTRAB is autonomous at the MS that carries out the detection, i.e., the detection method does not need information from other MSs. Furthermore, the MS builds the database portraying the normal protocol behavior profile, which is not dependent on the traffic volume. As a result of this design, the proposed detection scheme manages to avoid false alarms during flash crowd. The conducted simulations demonstrate the effectiveness of the detection technique. Our investigations have considered the attack detection delay and the "failed session detection error rate." We have also addressed the problem of tracing back attackers against encrypted protocols based on the correlated attack features at neighboring monitoring stubs.

As an approach of responding to the detected attacks, this work may be extended to selectively slow down the protocol response as long as the Cusum sequence exhibits anomalous behavior. Admittedly, when IPSEC protocol is employed by end-hosts through a secure tunnel, the transport layer headers may be encrypted and not visible to the MSs. Our future extensions to this work will consider how DTRAB may overcome such issues. The further extensions of our work may also facilitate combating against attacks on encrypted protocols in the wireless network environment.

References

[1] C. E. Landwehr and D. M. Goldschlag, "Security issues in networks with internet access," *Proc. IEEE*, vol. 85, no. 12, pp. 2034–2051, Dec. 1997.

[2] D. Bleichenbacher, "Chosen Ciphertext attacks against protocols based on the RSA encryption standard PKCS #1," in *Proc. 18th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, Aug. 1998, pp. 1–12.

[3] D. Brumley and D. Boneh, "Remote timing attacks are practical," in *Proc. 12th USENIX Security Symp.*, Washington, DC, Aug. 2003, p. 1.

[4] "OpenSSH PAM timing attacks," 2006 [Online]. Available: http://securityvulns.com/news2789.html

[5] S. P. Joglekar and S. R. Tate, "ProtoMon: Embedded monitors for cryptographic protocol intrusion detection and prevention," *J. Universal Comput. Sci.*, vol. 11, no. 1, pp. 83–103, Jan. 2005.

[6] Z. M. Fadlullah, T. Taleb, N. Ansari, K. Hashimoto, Y. Miyake, Y. Nemoto, and N. Kato, "Combating against attacks on encrypted protocols," in *Proc. IEEE ICC*, Glasgow, Scotland, Jun. 24–28, 2007, pp. 1211–1216.

[7] H. Wang, D. Zhang, and G. Shin, "Change-point monitoring for the detection of DoS attacks," *IEEE Trans. Depend. Secure Comput.*, vol. 1, no. 4, pp. 193–208, Oct.–Dec. 2004.

[8] J. P. Anderson, *Computer Security Threat Monitoring and Surveillance*. Fort Washington, PA: Anderson, 1980.

[9] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, and M. Embrechts, "Network-based intrusion detection using neural networks," in *Proc. ANNIE*, St. Louis, MO, Nov. 2002, pp. 10–13.

[10] "Smurf IP denial-of-service attacks," CERT Advisory CA-1998-01, 1998 [Online]. Available: http://www.cert.org/advisories/CA-1998-01.html

[11] "Ping of death," 1997 [Online]. Available: http://insecure.org/sploits/ping-o-death.html

[12] B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux, "Password interception in a SSL/TLS channel," in *Proc. Crypto 2003*, Santa Barbara, CA, Feb. 2003, vol. 2729, LNCS, pp. 583–599.

[13] D. Wagner and B. Schneier, "Analysis of the SSL 3.0 protocol," in *Proc. 2nd USENIX Workshop Electron. Commerce*, Oakland, CA, Nov. 1996, vol. 2, p. 4.

[14] "OpenSSL servers contain a buffer overflow during the SSL2 handshake process," CERT Vulnerability Note #102795, Jul. 2002.

[15] "OpenSSL servers contain a remotely exploitable buffer overflow vulnerability during the SSL3 handshake process," CERT Vulnerability Note #561275, Jul. 2002.

[16] F. Perriot and P. Szor, "An analysis of the slapper worm exploit," Symantec Security Response, Symantec White Paper, Apr. 2003.

[17] S. McClure and J. Scambray, "Once-promising intrusion detection systems stumble over a myriad of problems," *Info World*, vol. 22, p. 58, Dec. 2000.

[18] "SSLDump," [Online]. Available: http://www.rtfm.com/ssldump SSLDump

[19] A. Yamada, Y. Miyake, K. Takemori, A. Studer, and A. Perrig, "Intrusion detection for encrypted Web accesses," in *Proc. 21st Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Niagara Falls, ON, Canada, May 2007, pp. 569–576.

[20] A. Stavrou, D. L. Cook, W. G. Morein, A. D. Keromytis, V. Misra, and D. Rubenstein, "WebSOS: An overlay-based system for protecting Web servers from denial of service attacks," *Comput. Netw.*, vol. 48, no. 5, pp. 781–807, Aug. 2005.

[21] R. Oppliger, R. Hauser, and D. Basin, "SSL/TLS session-aware user authentication—OR how to effectively thwart the man-in-the-middle," *Comput. Commun.*, vol. 29, no. 12, pp. 2238–2246, Aug. 2006.

[22] J. Haggerty, Q. Shi, and M. Merabti, "Early detection and prevention of denial-of-service attacks: A novel mechanism with propagated traced-back attack blocking," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 10, pp. 1994–2002, Oct. 2005.

[23] D. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001, vol. 2, pp. 878–886.

[24] W. Lee and K. Park, "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001, vol. 1, pp. 338–347.

[25] S. Bellovin, "ICMP traceback messages," IETF Internet Draft, Mar. 2000.

[26] A. Belenky and N. Ansari, "On IP traceback," *IEEE Commun. Mag.*, vol. 41, no. 7, pp. 142–153, May 2003.

[27] T. Law, J. Lui, and D. Yau, "You can run, but you can't hide: An effective statistical methodology to trace back DDoS attackers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 9, pp. 799–813, Sep. 2005.

[28] Z. Gao and N. Ansari, "Tracing cyber attacks from the practical perspective," *IEEE Commun. Mag.*, vol. 43, no. 5, pp. 123–131, May 2005.

[29] H. Wu and S. S. Huang, "Detecting stepping-stone with Chaff perturbations," in *Proc. 21st Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Washington, DC, May 2007, pp. 85–90.

[30] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson, "Practical network support for IP traceback," in *Proc. SIGCOMM*, Stockholm, Sweden, Sep. 2000, pp. 295–306.

[31] X. Wang, D. S. Reeves, and S. F. Wu, "Inter-packet delay based correlation for tracing encrypted connections through stepping stones," in *Proc. 7th ESORICS*, Zurich, Switzerland, Oct. 2002, pp. 244–263.

[32] S. C. Lee and C. Shields, "Tracing the source of network attack: A technical, legal and societal problem," in *Proc. 2001 IEEE Workshop Inf. Assurance Security*, New York, Jun. 2001, pp. 239–246.

[33] A. Blum, D. Song, and S. Venkataraman, "Detection of interactive stepping stones: Algorithms and confidence bounds," in *RAID 2004*, E. Jonsson, A. Valdes, and M. Almgren, Eds., Heidelberg, 2004, vol. 3224, LNCS, no. 5, pp. 258–277.

[34] G. Mansfield, K. Ohta, Y. Takei, N. Kato, and Y. Nemoto, "Towards trapping wily intruders in the large," *Comput. Netw.*, vol. 34, no. 4, pp. 659–670, Oct. 2000.

[35] T. Taleb, Z. M. Fadlullah, K. Hashimoto, Y. Nemoto, and N. Kato, "Tracing back attacks against encrypted protocols," in *Proc. Int. Conf. Wireless Commun. Mobile Comput.*, Honolulu, HI, Aug. 12–16, 2007, pp. 121–126.

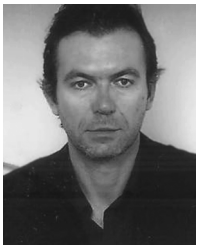
[36] "TCPDUMP/LIBPCAP public repository," [Online]. Available: http://www.tcpdump.org/
[37] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*. Englewood Cliffs, NJ: Prentice-Hall, Apr. 1993, pp. 25–26.
[38] S. Luo and G. A. Marin, "Realistic internet traffic simulation through mixture modeling and a case study," in *Proc. 2005 Winter Simulation Conf.*, Orlando, FL, Dec. 2005, pp. 2408–2416.
[39] "The network simulator: NS-2," [Online]. Available: http://www.isi.edu/nsnam/ns/
[40] "CAIDA: The Cooperative Association for Internet Data Analysis," [Online]. Available: http://www.caida.org/home/
[41] K. Sakaguchi, K. Ohta, Y. Waizumi, N. Kato, and Y. Nemoto, "Tracing DDoS attacks by comparing traffic patterns based on quadratic programming methods," (in Japanese) *Trans. IEICE B*, vol. J85-B, no. 8, pp. 1295–1303, 2002.
[42] "BRITE topology generator for NS-2," [Online]. Available: http://www.cs.bu.edu/brite/

Zubair M. Fadlullah (S'06) received the B.Sc. degree in computer sciences from the Islamic University of Technology, Dhaka, Bangladesh, in 2003 and the M.S. degree from the Graduate School of Information Sciences (GSIS), Tohoku University, Sendai, Japan, in March 2008. Currently, he is pursuing the Ph.D. degree at GSIS. His research interests are in the areas of network security, specifically intrusion detection/prevention, traceback, and quality-of-security service provisioning mechanisms.

Tarik Taleb (S'04–M'05) received the B.E. degree in information engineering with distinction and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively.

He is currently working as a Senior Researcher at NEC Europe Ltd., Heidelberg, Germany. Prior to his current position, he worked as an Assistant Professor at Tohoku University. His research interests lie in the field of architectural enhancements to 3GPP networks (i.e., LTE), mobile multimedia streaming, wireless networking, intervehicular communications, satellite and space communications, and network security.

Dr. Taleb is the recipient of many competitive awards such as the 2009 IEEE ComSoc Asia-Pacific Young Researcher Award, the 2008 TELECOM System Technology Award, and the 2006 IEEE Computer Society Japan Chapter Young Author Award. He has served as Vice Chair of the Satellite and Space Communications Technical Committee of the IEEE Communication Society (ComSoc) since 2006. He has also chaired a number of symposia in different IEEE conferences, including Globecom and ICC. He is on the Editorial Board of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, and a number of Wiley journals.



Athanasios V. Vasilakos (M'00) received the B.S. degree in electrical and computer engineering from the University of Thrace, Xanthi, Greece, in 1983; the M.S. degree in computer engineering from the University of Massachusetts, Amherst, in 1986; and the Ph.D. degree in computer engineering from the University of Patras, Patras, Greece, in 1988.

He is currently a Professor with the Department of Computer and Telecommunications Engineering, University of Western Macedonia, Kozani, Greece, and a Visiting Professor with the Graduate Program of the Department of Electrical and Computer Engineering, National Technical University of Athens (NTUA), Athens, Greece. He has authored or coauthored over 200 technical papers in major international journals and conferences. He is author/coauthor of five books and 20 book chapters in the areas of communications.

Prof. Vasilakos has served as General Chair, Technical Program Committee Chair, and symposium Chair for many international conferences. He is Chairman of the Intelligent Systems Applications Technical Committee (ISATC) of the IEEE Computational Intelligence Society (CIS). He served or is serving as an Editor or/and Guest Editor for many technical journals, such as the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, the IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the *IEEE Communications Magazine*, and the *ACM Transactions on Autonomous and Adaptive Systems*. He is founding Editor-in-Chief of the *International Journal of Adaptive and Autonomous Communications Systems* (IJAACS, http://www.inderscience.com/ijaacs) and the *International Journal of Arts and Technology* (IJART, http://www.inderscience.com/ijart).

Mohsen Guizani (F'09) received the B.S. (with distinction) and M.S. degrees in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, in 1984, 1986, 1987, and 1990, respectively.

He is currently a Professor and the Chair of the Information Science Department at Kuwait University, Safat, Kuwait. Prior, he was a Professor and Chair of the Computer Science Department at Western Michigan University, Kalamazoo. He is the author of six books and more than 200 publications in refereed journals and conferences. His research interests include computer networks, wireless communications and mobile computing, and optical networking.

Dr. Guizani is a Member of the IEEE Communication Society, the IEEE Computer Society, and the ASEE, and he is a Senior Member of the Association for Computing Machinery (ACM). He received both the Best Teaching Award and the Excellence in Research Award from the University of Missouri–Columbia in 1999 (a college-wide competition). He won the Best Research Award from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, in 1995 (a university-wide competition). He was selected as the Best Teaching Assistant at Syracuse University in 1988 and 1989. He currently serves on the editorial boards of six technical journals and is the Founder and Editor-in-Chief of *Wireless Communications and Mobile Computing* and the *Journal of Computer Systems, Networks and Communications*. He guest edited a number of special issues in IEEE journals and magazines. He is also the Founder and General Chair of the IEEE International Conference of Wireless Networks, Communications, and Mobile Computing (IWCMC). He has served as member, Chair, and General Chair of a number of conferences. He is the Past Chair of TAOS and current Chair of WTC IEEE ComSoc Technical Committees.

Nei Kato (M'03–SM'05) received the M.S. and Ph.D. degrees in information engineering from Tohoku University, Sendai, Japan, in 1988 and 1991, respectively.

He has been a Full Professor with the Graduate School of Information Sciences (GSIS), Tohoku University, since 2003.

Prof. Kato is the recipient of such awards as the 2005 Distinguished Contributions to Satellite Communications Award from the IEEE Communications Society, the 2009 IEICE Network System Research Award, and several other foundation awards. He has served as a symposium Co-Chair for IEEE GLOBECOM 2007 and IEEE ICC 2010. He currently serves as Vice Chair of the Technical Committee of Satellite Communications, IEICE; a Technical Editor of *IEEE Wireless Communications* (2006–present), Associate Editor of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY (2009–present), and an Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (2008–present).