Fast Detection of Denial-of-Service Attacks on IP Telephony

Hemant Sengar† Haining Wang ‡ Duminda Wijesekera† Sushil Jajodia†

†Center for Secure Information Systems
George Mason University
Fairfax, VA 22030
{*hsengar,dwijesek,jajodia*}*@gmu.edu*

‡Department of Computer Science
College of William and Mary
Williamsburg, VA 23187
hnw@cs.wm.edu

Abstract— **Recently Voice over IP (VoIP) is experiencing a phenomenal growth. Being a real-time service, VoIP is more susceptible to Denial-of-Service (DoS) attacks than regular Internet services. Moreover, VoIP uses multiple protocols for call control and data delivery, making it vulnerable to various DoS attacks at different protocol layers. An attacker can easily disrupt VoIP services by flooding TCP `SYN` packets, UDP-based RTP packets, or SIP-based `INVITE` messages, which pose a critical threat to IP telephony. In this paper, we present an online statistical detection mechanism, called vFDS, to detect DoS attacks in the context of VoIP. The core of vFDS is based on *Hellinger distance* method, which computes the variability between two probability measures. Using *Hellinger distance*, we characterize normal protocol behaviors and then detect the traffic anomalies caused by flooding attacks. Our experimental results show that vFDS achieves fast and accurate detection of DoS attacks.**

I. Introduction

IP telephony, commonly known as *Voice over IP* (VoIP), provides a viable alternative to traditional wired line and wireless telephone systems. As its deployment spreads, VoIP will become a prime target for DoS attacks, in which flooding attack is the most common one due to the readily available tools and its simple nature. In this study, we focus on those DoS attacks of simply flooding packets. Such DoS attacks pose a serious threat to IP telephony infrastructure. They deteriorate the perceived quality of service (QoS) and even cripple down the devices in the path from caller to callee, such as IP telephones, SIP proxy servers, and softswitches.

Unlike other Internet services, VoIP uses multiple protocols for call control and data delivery. For example, in SIP-based IP telephony, Session Initiation Protocol (SIP) is used to control call setup and teardown, while Real-time Transport Protocol (RTP) is used for voice delivery. Additionally, a VoIP system is distributed in nature, including IP phones, SIP proxies, and many other servers, and is susceptible to both transport layer (TCP) and application layer (SIP, RTP) flooding attacks. Consequently, defending against such a wide range of DoS attacks requires a generic defense mechanism working across different protocol layers.

Utilizing the Sequential Change Point Detection scheme [5], Application Layer Attack Sensor (ALAS) [20] defends IP telephony against DoS attacks. The same method has been applied to detect TCP SYN flooding attacks [27]. These detection mechanisms are based on the observation that a DoS attack causes a large number of incomplete handshaking processes in either SIP or TCP. If there is a sudden surge of such incomplete handshaking processes, then it is an indication of DoS attacks. ALAS [20] relies on the difference between `INVITE` and `200 OK` message pairs to detect the start of an `INVITE` flooding attack. Our work differs from ALAS in many aspects. First, the {`INVITE`, `200 OK`} message pair works well for detecting an `INVITE` flooding source inside the enterprise network. However, its detection of flooding attacks originated outside the enterprise network is questionable, since the pair discrepancy occurs only after the flooding traffic has exhausted the victim's resources. Second, the call setup phase is completed by a three-way handshake of `INVITE/200 OK/ACK` messages. The observation of an `INVITE` and `200 OK` message pair only confirms the *dialog*, not the *session* establishment. A session is established only after receiving an `ACK` message. Moreover, the most serious threats to IP telephony are posed by RTP (i.e. media) flooding attacks that do not involve any handshaking, and hence, cannot be detected by the above mentioned mechanisms.

The stochastic nature of phone call arrival varies with the change of time, and cannot be easily modeled by a random process with a deterministic time-varying arrival rate [9], [14], making it a challenge to develop a flooding attack signature for IP telephony. Therefore, instead of characterizing the complex call request behaviors, our flooding detection mechanism is based on the inherent protocol behaviors and related message types. For example, there is a strong correlation between SIP's `INVITE`, `200 OK` and `BYE(CANCEL)` messages, and similarly between TCP's `SYN`, `SYN-ACK` and `FIN(RST)` packets. Any deviation from this kind of correlation is a strong indication of DoS attacks.

The previous protocol-behavior-based solutions [20], [27], [28] are limited in their capabilities. They operate on the difference between the selected protocol attribute pairs. For example, Reynolds et al. [20] used the difference between {`INVITE`, `200 OK`} and {`SYN`, `ACK`} attribute pairs for detecting `INVITE` and `SYN` flooding attacks, respectively. Wang et al. [27] used {`SYN`, `FIN`} pair for `SYN` flooding attack

1-4244-0476-2/06/$20.00 ©2006 IEEE.

detection. These individual pairs do not present a holistic view of protocol behaviors and are susceptible to mixed traffic (i.e. INVITE with 200 OK or SYN with FIN) flooding attacks. Moreover, in the case of IP telephony, an RTP-based media stream does not have any such observable protocol behaviors. Now the questions arise:

- *Is it possible to detect all three (i.e. SYN, INVITE and RTP) flooding attacks?*
- *How to detect these flooding attacks in a generic way, instead of running separate mechanism for each one?*
- *Is there a simple way to characterize complex protocol behaviors based on protocol attributes?*

To answer these questions, we propose a *Hellinger distance* based VoIP flooding detection system (vFDS). vFDS is a simple online statistical detection mechanism. To validate its effectiveness, we evaluate vFDS using Internet traces and VoIP traffic traces. The Internet traces are collected from the exchange points that connect stub networks to the Internet. VoIP traces are generated from a realistic SIP-based IP telephony testbed. Our experimental results demonstrate that vFDS can achieve high detection accuracy with short detection time. Note that the Hellinger distance-based detection approach itself is quite generic, and can be easily extended to detect other network anomalies beyond the context of VoIP.

The remainder of this paper is structured as follows. Section II briefly describes the background of this study. Section III presents the threat model. Section IV analyzes normal traffic behaviors and discusses the inherent correlation among protocol attributes. Section V describes Hellinger distance and shows how to apply it for characterizing traffic streams. Section VI evaluates the performance of vFDS. Section VII surveys related work. Finally, we conclude the paper with Section VIII.

II. Background

A. SIP-based IP Telephony

As the standard signaling protocol for VoIP, SIP is a text-based application level protocol to set up, modify, and tear down multimedia sessions between one or more participants. There are two basic types of components in SIP, *user agents* (UAs) (i.e. IP phones) and *SIP servers* (i.e. Location, Redirect, Registrar and Proxy servers). Each UA is a combination of two entities, the *user agent client* (UAC) and the *user agent server* (UAS). The UA switches back and forth between being a UAC and a UAS.

The SIP messages are classified into two groups: *requests* and *responses*. The SIP requests are also called methods, and there are six of them (INVITE, ACK, BYE, CANCEL, REGISTER and OPTIONS) described in [21]. Other methods are proposed as the extensions of the original six methods. For each request of a UAC, SIP server (or UAS) generates a SIP response. Each response message is identified by a numeric status code.

Now, we give an example of a typical call setup flow to highlight the usage of SIP request and response messages between user agents UA-A and UA-B. Suppose that the two UAs belong to different domains, which have their own proxy servers. UA-A calls UA-B using its SIP phone over the Internet. The outbound proxy server uses the Domain Name System to locate the inbound proxy server at the other domain. After obtaining the IP address of the inbound proxy server, the outbound proxy server of UA-A sends the INVITE request to the domain of UA-B. The inbound proxy server consults a location service database to find out the current location of UA-B, and forwards the INVITE request to the UA-B's SIP phone. Exchanging INVITE, 200 OK and ACK messages completes the three-way handshake and establishes a SIP session. A set of parameters are exchanged via SIP messages (in the message body using Session Description Protocol (SDP) [12]) between the two end points before a RTP-based voice channel is established. In general, the path of media packets is independent of that of the SIP signaling messages. At the end of the call, UA-B (or UA-A) hangs up by sending a BYE message. Subsequently, UA-A (or UA-B) terminates the session and sends back a 200 OK response. This example shows the basic functionality of SIP, and the detailed description of the SIP operations is in RFC 3261 [21].

B. Enterprise IP Telephony Network and Placement of vFDS

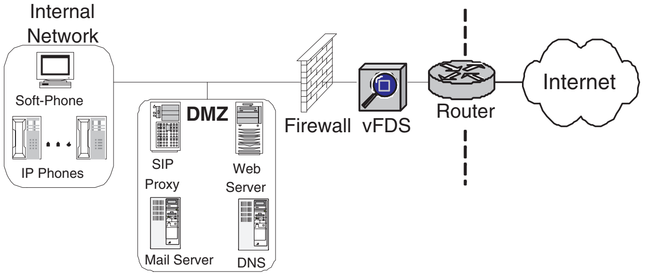
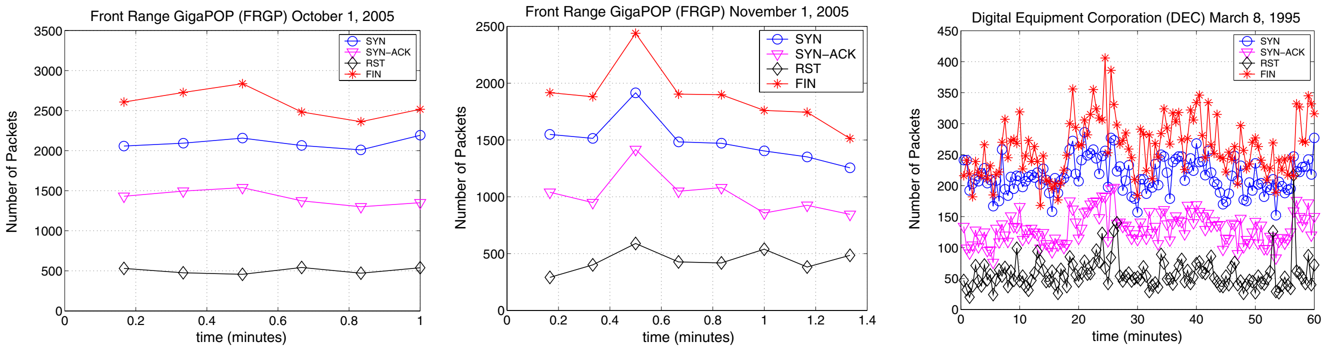


Fig. 1. Enterprise IP Telephony Network

An enterprise network consists of internal network and demilitarized zone (DMZ). DMZ contains many servers including a SIP proxy server. Note that SIP proxy servers have no media capabilities and only facilitate the two end points (i.e. IP telephones) to discover and contact each other through SIP signaling. Once the end points have been located, the media flows directly from end to end without going through a proxy. Therefore, vFDS requires the visibility of crossing traffic to monitor the signaling messages and media flows for all SIP clients.

Under the assumption that most VoIP related security threats arise outside the enterprise network, the online vFDS is located strategically between the edge router and the firewall, allowing the visibility of all traffic traveling to and from both DMZ and the internal network to the Internet. The placement of vFDS also obviates the need for flooding detection mechanism at every single SIP entity (i.e. PCs, phones, servers). Figure 1 shows a SIP-based IP telephony enabled enterprise network and the placement of vFDS. Since vFDS operates on the traffic streams at transport and application layers, in our experiments



(a.) FRGP-1 (b.) FRGP-2 (c.) DEC-1

Fig. 2. TCP Attribute Behaviors in Various Traces

we use Netfilter [17] for traffic filtering and ethereal [2], which is SIP and RTP (application layer protocols) aware, for traffic analysis.

III. The Threat Model

In this section, we describe various DoS attacks at different protocol layers of IP telephony services. Our work is mainly focused on detecting SYN, INVITE and RTP-based flooding attacks.

A. Transport layer attacks

The transport layer is responsible for transmission of SIP signaling messages and media streams. The RTP-based media streams are based on UDP, whereas the SIP signaling messages can be carried by TCP, SCTP or UDP. Both TCP and UDP based clients are susceptible to flooding attacks. UDP being a stateless protocol is less vulnerable compared to TCP. However, SIP's attempt to provide transport reliability by maintaining state, timers and acknowledgments makes it susceptible to UDP flooding attacks as well.

B. Application layer attacks

SIP-based entities (e.g. PCs, laptops, IP phones, and SIP servers etc.) are susceptible to two types of flooding attacks. In the first type of attacks, bogus traffic is directed towards a SIP entity with the aim of exhausting its CPU resource or the bandwidth of the connecting link. In the second type of attacks, an attacker exploits the vulnerabilities in the SIP protocol itself. SIP is a transactional protocol, where each transaction consists of a request and its corresponding response. *Transaction users* (TUs) at a SIP entity (except stateless proxy) maintain a transaction state for some time. An INVITE transaction is particularly susceptible to a flooding attack, since it may take several seconds (or minutes) to complete. For example, a SIP proxy has the option to maintain an INVITE transaction state up to 3 minutes. Similarly, when a UAS accepts the INVITE request, TU generates a 2XX response and waits for an ACK while maintaining the transaction state. In all these and other examples, the TU's finite capacity of maintaining state for the transaction could easily be exploited by flooding spurious requests.

RTP delivers real-time media from one end point to another. An RTP flooding attack exploits the vulnerabilities of the media path to deteriorate the perceived voice quality. In the RTP flooding attack, the attacker sends a barrage of fabricated RTP packets without following any media encoding scheme. This attack attempts to exhaust the available bandwidth and even make IP phones dysfunctional.

IV. Normal Traffic Behaviors

Based on the real Internet traces and the VoIP traces obtained from our testbed, we profile the normal behaviors of protocol attributes, which are defined as the packet (message) types appeared in the traffic traces. For example, TCP's control packets, SIP's request and response messages are considered as protocol attributes. We observe that although the nature of Internet (including VoIP) traffic is quite diverse and independent of each other, the inherent correlation among protocol attributes is more or less well maintained. Quantifying this inherent correlation among protocol attributes, we may differentiate attacking traffic behavior from normal traffic behavior.

A. Behavior of TCP attributes

To study the TCP attribute behaviors, we choose two sets of traces representing real-life Internet traffic at the exchange points. These exchange points connect the stub networks to the Internet. The collection times of the two sets of traces are ten years apart, which are deliberately chosen to demonstrate the invariant nature of the TCP protocol attribute behaviors, irrespective of the ever-changing Internet traffic. The first set of traces were gathered from the *Front Range GigaPOP* (FRGP) [16], in which one trace (FRGP-1) was collected on Saturday, October 1, 2005, and the other trace (FRGP-2) was collected on Tuesday, November 1, 2005. The second set of trace is the collection of one hour's worth WAN traffic between *Digital Equipment Corporation* (DEC) [8] and the rest of the Internet. The trace ran from 22:00 to 23:00 on Wednesday, March 8, 1995. Note that all the traces are bi-directional. We parse the traces and extract SYN, SYN-ACK, FIN and RST packets from the TCP streams.

Figure 2 illustrates the behaviors of SYN, SYN-ACK, FIN and RST attributes of the TCP streams. In the normal TCP handshake process, for each SYN request from the client, there is one SYN-ACK response from the server. However, as shown in Figure 2, the strict one-to-one mapping between SYN and SYN-ACK is not observed. The curve of SYN is clearly above that of SYN-ACK. The plausible reasons for this discrepancy include SYN losses and its retransmission, and the cases in which a server is either down or heavily overloaded so that no SYN-ACK is generated for each received SYN packets. Also, under normal conditions, a TCP connection started by a SYN packet will ultimately be torn down by the exchange of two FIN packets (considering back and forth exchange between the client and the server). Figure 2 shows that the FIN curve lies above the SYN curve but not always at twice the height of the SYN-ACK curve. The discrepancy can be attributed to the following reasons: not every observed SYN-ACK leads to an established TCP connection and a RST packet can terminate an established TCP connection without generating any FIN packets.

B. Behavior of SIP attributes

In order to study the attribute behaviors of VoIP traffic, we build a testbed including SIP proxy servers and SIP-based softphones. The testbed consists of four PCs (500 MHz Pentium III CPU, 128 Mbytes RAM) equipped with Linux operating system acting as SIP clients, SIP servers and routers. Figure 3 illustrates the layout of the testbed, in which we generate VoIP traffic and evaluate the performance of vFDS. Enterprise networks A and B are simulated by two different PCs equipped with SIP traffic generators, which can behave as multiple UACs trying to make calls to UASs in enterprise network C.

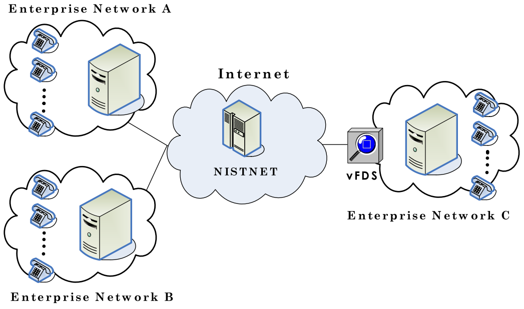


Fig. 3. Layout of SIP-based VoIP Testbed

The average call generation rate is 50 calls per second, with the lowest call rate of 25 calls per second and the peak call rate of 70 calls per second. The talking time is set to 60 seconds. The voice codec algorithm used is $G.711$ (50 packets per second). The Wide Area Network (WAN) emulator ("NISTNet" [6]) connects networks A and B to network C using 100 Mbps Ethernet links. The NIST package runs on a Linux router, in which packet delay distribution, congestion, loss, and bandwidth are configurable. We set the Internet delay to 50 ms and the packet loss rate to 0.42% in our experiments. We use the Network Time Protocol (NTP) to synchronize the time of clients with that of the NISTNET server. The SIP signaling messages are carried by UDP. SIP timer $T1$ is set to 500 ms. The experiment runs for an hour.

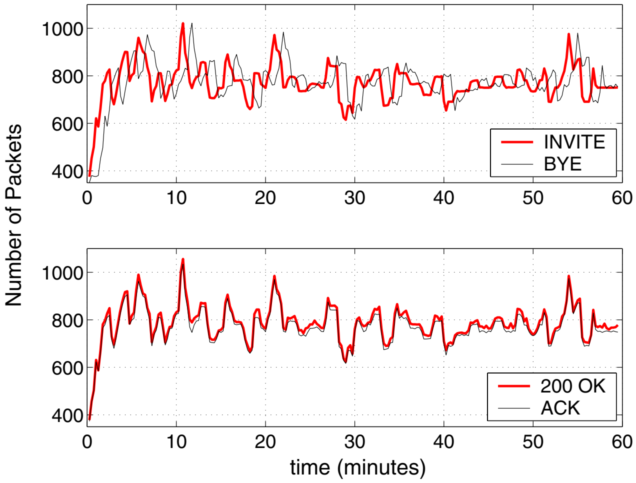


Fig. 4. SIP Attribute Behaviors

Figures 4 plots the observed SIP attribute behaviors at the SIP proxy server of enterprise network C. As shown in Figure 4, the 200 OK and ACK curves are closely overlapped with each other, while there are occasional small gaps between them. During the one hour run, we observe $3,545$ 200 OK and 790 BYE retransmissions. In addition, there are 109 time-outs. Because of these time-outs and retransmissions, the strict one-to-one mapping between INVITE and other SIP messages such as 200 OK, ACK and BYE is violated. However, the strong positive correlations do exist between INVITE, 200 OK, ACK and BYE messages.

C. Behavior of RTP attributes

As mentioned earlier, compared to TCP and SIP, RTP does not have any observable protocol behavior. At the application layer, we can only observe the number of RTP packets received per time unit. Based on the number of RTP packets, we define two attributes $n_{theoretical}$ and $n_{observed}$, for a virtual RTP stream and an observable RTP stream, respectively. The caller's media stream attribute $n_{theoretical}$ provides a base for comparison with $n_{observed}$ (i.e. the observed number of packets in real RTP stream). Both attributes represent the number of packets in a given time interval. At the application layer, vFDS can easily determine the value of $n_{observed}$ by counting the number of incoming RTP packets for each voice stream, which is uniquely identified by the combination of the destination IP address and port number. To determine the value of $n_{theoretical}$, we need to incorporate the communication between SIP and RTP protocol state machines [22]. vFDS can fulfill this requirement, since it monitors the signaling messages and media packets of each call. The call control (SIP) and media delivery (RTP) protocols are synchronized by exchanging the synchronization messages for critical events in the established sessions. The media attributes such as

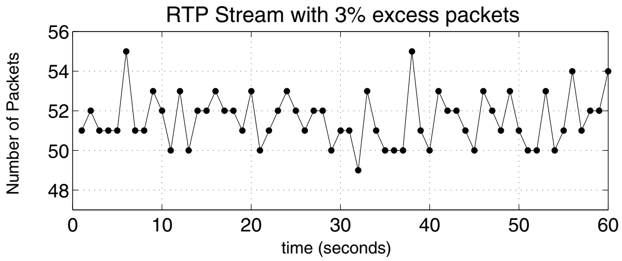


Fig. 5. RTP Attribute Behaviors

media_format, media_encoding, and sampling_rate, which are included in SIP message body, are accessible to the RTP state machine via the SIP state machine. Suppose that the media information for caller is ⟨v.media_format = audio, v.media_encoding = PCMU (i.e. G.711), v.sampling_rate = 8000⟩. Then, the number of packets per second (PPS) without enabling voice activity detection is 50, and the voice payload size is 160 Bytes with the codec sample interval of 10 ms. Thus, the value of $n_{theoretical}$ attribute for this particular media stream is 50 PPS.

Inside the enterprise network, all these individual media attributes for the callers can be integrated together, instead of keeping track of $n_{theoretical}$ attribute for each caller's RTP stream individually.

$$n_{\texttt{theoretical}} = \sum_{i=1}^{N} n^{i}_{\texttt{theoretical}}$$

In the equation above, during $\triangle t$ time period, there are $n^{i}_{\texttt{theoretical}}$ number of packets in virtual stream i, where N represents the total number of open virtual streams. That is, there are N active calls, each with its own incoming RTP stream and each stream with its own negotiated media encoding scheme.

In our experiments, to observe RTP attribute behaviors, we assume that UACs use G.711 (i.e $n_{theoretical} = 50$ PPS) codec algorithm. Figure 5 shows one instance of the simulated RTP stream trace with 3% duplicate (i.e. excess) packets. In this example, we have considered only one incoming media RTP stream, while it could be generalized to include any number of RTP streams.

D. Inherent Attribute Correlation

In the collected traces, the ideal behavior of one-to-one mapping between protocol attributes is not always held. The observed discrepancy is due to prevailing network conditions such as packet droppings and retransmissions. However, in spite of traffic diversity, at any instant of time, the strong correlations between protocol attributes are clearly held in traces. The distances between attributes (i.e. intrinsic correlation) do not vary much with the change of time. and have an observable correlation with the total number of packets.

Given any two different instants of time, we attempt to quantify the observable similarity (or dissimilarity) of these two instants. The similarity quantification provides a way to validate an unknown observed traffic against the profiled normal traffic. If the attribute behavior of the observed traffic is similar to that of the known normal traffic, then it is classified as legitimate traffic; otherwise, it is suspected to be an instance of an attack. For this validation to be successful, it is necessary to profile the normal traffic first. The framework of our detection scheme has two periods: a *training period* and a *testing period*. During the training period, we characterize the protocol attribute correlations of the normal traffic. The distances between attributes in the normal traffic will be used as a reference later in the testing period, to compare with the distances of attributes in the observed traffic. In the next section, we show the quantitative characterization of protocol attributes in the training and testing periods.

V. Statistical Attack Detection

Intrusion detection techniques are generally divided into two paradigms, *anomaly detection* and *misuse detection*. Misuse detection is based on the matching of attack signatures, whereas anomaly detection is based on the deviation from normal system behaviors. vFDS can be viewed as a statistical anomaly detection system. Each user's call behavior as well as the accumulative behavior of all users inside an enterprise IP telephony network are profiled, and any significant deviation from the normal behavior is an indication of intrusion activity. In the following, we describe Hellinger distance and develop a distance measurement technique.

A. Hellinger Distance

Hellinger distance presents an intrinsic way to estimate the distances between probability measures independent of the parameters. It is closely related to the *total variation distance* but with several advantages. Let $\mathbb{P}$ and $\mathbb{Q}$ be the two probability measures on a finite event space Ω. Probability measure $\mathbb{P}$ on Ω is an N-tuple $(p_1, p_2, ..., p_N)$, which satisfies $p_\alpha \geq 0$ and $\sum_\alpha p_\alpha = 1$. Similarly, $\mathbb{Q}$ on Ω is also an N-tuple $(q_1, q_2, ..., q_N)$, which satisfies $q_\alpha \geq 0$ and $\sum_\alpha q_\alpha = 1$. The *Hellinger distance* between $\mathbb{P}$ and $\mathbb{Q}$ is defined as :

$$d_H^2(\mathbb{P}, \mathbb{Q}) = \frac{1}{2} \sum_{\alpha=1}^{N} (\sqrt{p_\alpha} - \sqrt{q_\alpha})^2$$

The Hellinger distance satisfies the inequality of $0 \leq d_H^2 \leq 1$. The distance is 0 when $\mathbb{P} = \mathbb{Q}$. Disjoint $\mathbb{P}$ and $\mathbb{Q}$ shows the maximum distance of 1. Sometimes, the factor $\frac{1}{2}$ is not used in the above equation. Further details on *Hellinger distance* can be found in [18], [10].

B. Distance Measurement

To detect a protocol behavior violation, we only need a fraction of specific attributes. The selection of attributes not only depends upon the protocol type but also on the violation we want to detect. In the following examples, we choose N attributes of a protocol, which satisfy $p_\alpha, q_\alpha \geq 0$ and $\sum_\alpha p_\alpha = 1$, $\sum_\alpha q_\alpha = 1$. Here α represents an attribute in the chosen set of N attributes. Probability measure $\mathbb{P}$ is defined over the training data set, whereas probability measure $\mathbb{Q}$ is defined over the testing data set. Both $\mathbb{P}$ and $\mathbb{Q}$ are hypothesized to be an array of normalized frequencies of all N attributes.

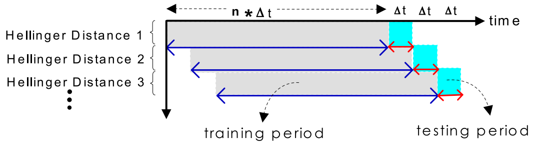


Fig. 6. Relationship between Training and Testing periods

As shown in Figure 6, the training data set is collected over n sampling periods over a normal traffic stream. The duration of each period is $\triangle t$. This initial training data set is devoid of any attack and acts as a base for comparison with the next $(n+1)^{th}$ period of the testing data set. Using the *Hellinger distance* approach, we measure the similarity between these two data sets. If the measured distance exceeds a threshold, then an alarm is raised to network administrators. If the distance is below the threshold, then the testing data set is included into the previous $(n-1)$ sampled traffic data to derive a new training data set. This moving window mechanism helps the training data set to adapt with the dynamics of network traffic. In the next few examples, we measure the *Hellinger distances*, under the various combinations of protocols and their attributes.

1) Example One: In the first example, at the TCP layer, we choose four attributes SYN, SYN-ACK, FIN, and $\texttt{RST}_{active}$. To filter out $\texttt{RST}_{passive}$ from the observed RSTs, we applied the threshold filter of [27]. Henceforth in this paper, we will not distinguish between RST and $\texttt{RST}_{active}$ packets. Now, suppose there are $N_{\texttt{SYN}}$, $N_{\texttt{SYN-ACK}}$, $N_{\texttt{FIN}}$, and $N_{\texttt{RST}}$ packets during the training period (i.e. in $n * \triangle t$ time). $\mathbb{P}$ is an array of normalized frequencies of $p_{\texttt{SYN}}$, $p_{\texttt{SYN-ACK}}$, $p_{\texttt{FIN}}$, and $p_{\texttt{RST}}$ over the training data set.

$$p_\alpha = N_\alpha / N_{Total}, \text{ where } \alpha \in \{\texttt{SYN}, \texttt{SYN-ACK}, \texttt{FIN}, \texttt{RST}\}$$
$$\text{and} \quad N_{Total} = (N_{\texttt{SYN}} + N_{\texttt{SYN-ACK}} + N_{\texttt{FIN}} + N_{\texttt{RST}})$$

During the testing period (i.e. at the $(n+1)^{th}$ sampling duration), $\mathbb{Q}$ is an array of normalized frequencies of $q_{\texttt{SYN}}$, $q_{\texttt{SYN-ACK}}$, $q_{\texttt{FIN}}$, and $q_{\texttt{RST}}$. In this time period, we observe $N^{\backprime}_{\texttt{SYN}}$, $N^{\backprime}_{\texttt{SYN-ACK}}$, $N^{\backprime}_{\texttt{FIN}}$ and $N^{\backprime}_{\texttt{RST}}$ packets.

$$q_\alpha = N^{\backprime}_\alpha / N^{\backprime}_{Total}, \text{ where } \alpha \in \{\texttt{SYN}, \texttt{SYN-ACK}, \texttt{FIN}, \texttt{RST}\}$$
$$\text{and} \quad N^{\backprime}_{Total} = (N^{\backprime}_{\texttt{SYN}} + N^{\backprime}_{\texttt{SYN-ACK}} + N^{\backprime}_{\texttt{FIN}} + N^{\backprime}_{\texttt{RST}})$$

To calculate the *Hellinger distance* (HD) between $\mathbb{P}$ and $\mathbb{Q}$ at the end of $(n+1)^{th}$ sampling period, we use

$$HD_1 = (\sqrt{p_{\texttt{SYN}}} - \sqrt{q_{\texttt{SYN}}})^2 + (\sqrt{p_{\texttt{SYN-ACK}}} - \sqrt{q_{\texttt{SYN-ACK}}})^2 + (\sqrt{p_{\texttt{FIN}}} - \sqrt{q_{\texttt{FIN}}})^2 + (\sqrt{p_{\texttt{RST}}} - \sqrt{q_{\texttt{RST}}})^2$$

Similarly, to calculate the HD for the next sampling period (i.e. at the end of $(n+2)^{th}$ period), we use previous n (from 2 to $(n+1)^{th}$) sampling period data as the training data and $(n+2)^{th}$ period data for the testing purpose.

The top of Figure 7 illustrates the *Hellinger distance* of the DEC trace including all four attributes at the same time.

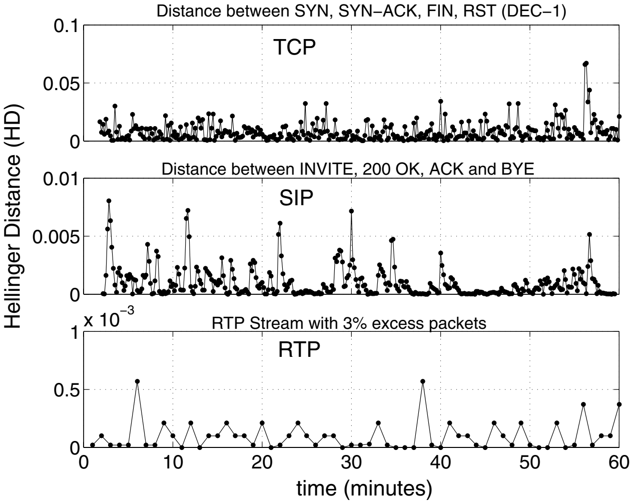


Fig. 7. Hellinger Distance of TCP, SIP and RTP Attributes

Throughout the one hour duration, the TCP attribute behaviors of the DEC trace demonstrate a remarkable similarity with time, given the fact that the *Hellinger distance* of zero (i.e. HD = 0.0) represents the same probability measures. The DEC trace sample has an average low distance of 0.007 and the maximum distance observed is 0.064.

2) Example Two: In the second example, at the application layer, we choose SIP protocol attributes INVITE, 200 OK, ACK and BYE. Here, the probability measure $\mathbb{P}$ is an array of the normalized frequencies of $p_{\texttt{INVITE}}$, $p_{\texttt{200 OK}}$, $p_{\texttt{ACK}}$ and $p_{\texttt{BYE}}$ over the training data set. Similarly, $\mathbb{Q}$ is an array of $q_{\texttt{INVITE}}$, $q_{\texttt{200 OK}}$, $q_{\texttt{ACK}}$ and $q_{\texttt{BYE}}$ during the testing period. All other details are similar to the previous example. To calculate the *Hellinger distance* (HD) between $\mathbb{P}$ and $\mathbb{Q}$, we use

$$HD = (\sqrt{p_{\texttt{INVITE}}} - \sqrt{q_{\texttt{INVITE}}})^2 + (\sqrt{p_{\texttt{200 OK}}} - \sqrt{q_{\texttt{200 OK}}})^2 + (\sqrt{p_{\texttt{ACK}}} - \sqrt{q_{\texttt{ACK}}})^2 + (\sqrt{p_{\texttt{BYE}}} - \sqrt{q_{\texttt{BYE}}})^2$$

The middle of Figure 7 shows the Hellinger distance for the SIP protocol attributes set of $\{\texttt{INVITE}, \texttt{200 OK}, \texttt{ACK}, \texttt{BYE}\}$. The maximum distance observed is $8 * 10^{-3}$ and the average distance for the entire run is $0.9 * 10^{-3}$. Such a low value of Hellinger distance indicates the closeness between the observed and training traffic behaviors.

3) Example Three: In the third example, at the application layer, we choose RTP protocol and its derived attributes $n_{theoretical}$ and $n_{observed}$. The probability measure $\mathbb{P}$ at time $t = 0$ is:

$$p_{\texttt{theo.}} = n_{\texttt{theoretical}} / (n_{\texttt{theoretical}} + n_{\texttt{observed}})$$
$$p_{\texttt{obs.}} = n_{\texttt{observed}} / (n_{\texttt{theoretical}} + n_{\texttt{observed}})$$

where both $n_{theoretical}$ and $n_{observed}$ are initialized to 50 PPS and thus giving the values of $p_{\texttt{theo.}} = p_{\texttt{obs.}} = 1/2$. $\mathbb{P}$ remains constant for the subsequent sampling periods, except when it is changed by a SIP's re-INVITE message (i.e. change of media encoding scheme). $\mathbb{Q}$ for each testing period $\triangle t$ is calculated as :

$$q_{\texttt{theo.}} = n_{\texttt{theoretical}} * \triangle t / (n_{\texttt{theoretical}} * \triangle t + n_{\texttt{observed}})$$
$$q_{\texttt{obs.}} = n_{\texttt{observed}} / (n_{\texttt{theoretical}} * \triangle t + n_{\texttt{observed}})$$

$n_{\texttt{observed}}$ is the actual number of RTP packets observed for a particular voice stream during $\triangle t$ time period. At the end of the first sampling period, the Hellinger distance for Caller's media stream is computed as

$$HD_1 = (\sqrt{p_{\texttt{theo.}}} - \sqrt{q_{\texttt{theo.}}})^2 + (\sqrt{p_{\texttt{obs.}}} - \sqrt{q_{\texttt{obs.}}})^2$$

For the subsequent testing periods, the computation of Hellinger distance is the same as the above by only changing the values of $q_{\texttt{theo.}}$ and $q_{\texttt{obs.}}$ for that particular $\triangle t$. The bottom of Figure 7 shows the observed Hellinger distances for a RTP stream trace with 3% duplicate packets. The observed distances are in the order of $\simeq 10^{-4}$.

C. Data Classification and Sampling

The traffic analysis and packet classification are performed at the transport layer first, and then at the application layer. The data sampling duration at both protocol layers is $\triangle t$. The size of this sampling window determines the detection resolution of flooding attacks and the computational overhead of vFDS. Most of TCP connections last for 12 to 19 seconds [26], whereas IP phone calls last much longer, 50% calls complete around one minute and 10% calls last even longer than 10 minutes [25]. In order to correlate a `SYN` with the `FIN(RST)` of the same connection and an `INVITE` with the corresponding `BYE`, the sampling window size needs to be 19 seconds and one minute, respectively. Fortunately, our detection mechanism is not sensitive to per-flow state information. It is based on the correlation between the aggregated `SYNs` to the corresponding `FINs(RSTs)` and the aggregated `INVITEs` to the corresponding `BYE(CANCEL)` messages. In our detection scheme, we set the sampling period to 10 seconds to achieve high detection resolution and relatively low CPU overhead. In addition to the sampling period $\triangle t$, *Hellinger distance* measurement also depends upon the training period ($n * \triangle t$). A longer training period provides more accurate (i.e. Hellinger distance) anomaly detection during the testing period, whereas a shorter training period adapts to the dynamics of network traffic more quickly. To balance the accuracy and responsiveness, we set the training period to 120 seconds (i.e. $n = 12$ samples) in all of the traces.

D. Detection Threshold

Using Hellinger distance, we have profiled the normal protocol attribute behaviors. Note that the distance between $\mathbb{P}$ and $\mathbb{Q}$ is defined as : $HD = \sum_{\alpha=1}^{N} (\sqrt{p_\alpha} - \sqrt{q_\alpha})^2$, i.e., the distance consists of the sums of square terms. In other words, all attributes (α) of a protocol contribute towards the value of the distance. Therefore, the distance can capture the holistic protocol behaviors. Any attribute deviation from the normal behavior is reflected in the Hellinger distance measurement.

Now our goal is to seek a threshold of Hellinger distance that can clearly differentiate network anomalies from normal behaviors. Instead of choosing a *static threshold* value, we estimate a *dynamic threshold* to capture the essence of the dynamic nature of VoIP traffic streams. The threshold distance for the testing period is based on the previously-measured *Hellinger distance* and the *stochastic gradient algorithm*.

The estimation of the threshold distance is an instance of Jacobson's *Fast algorithm for RTT mean and variation* [13]. Fast estimators for average a and mean deviation ν, given measurement m, can be computed as :

$$Err = m_n - a_{n-1} \quad (1)$$
$$a_n \longleftarrow a_{n-1} + g.Err \quad (2)$$
$$\nu_n \longleftarrow \nu_{n-1} + h.(|Err| - \nu_{n-1}) \quad (3)$$

where m_n is the current sample of the Hellinger distance, a_{n-1} and a_n are the previous and current smoothened Hellinger distances, respectively, ν_{n-1} and ν_n represent the previous and current mean deviations. To make the computation efficient, g and h are chosen to be inverses of powers of 2. Here we set $g = \frac{1}{2^3}$ and $h = \frac{1}{2^2}$ as previous research suggested [23]. The smoothened Hellinger distance a_n (in eq. 2) is based on the observed Hellinger distance m, which is measured between the probability measures $\mathbb{P}$ and $\mathbb{Q}$ (see Section V-B). During the testing periods, we derive the estimated threshold Hellinger distance ($HD^{thresh.}$) using the smoothened Hellinger distance (eq. 2) and the mean deviation (eq. 3).

$$HD_{n+1}^{thresh.} = \underbrace{X * a_n} + \underbrace{\eta * \nu_n} \quad (4)$$

The purpose of the multiplication factors X and η is to give a safe margin for the setting of the threshold value, so that vFDS avoids any false alarms without degrading its detection sensitivity. The first factor in Equation (4), which largely depends upon the observed Hellinger distances, should be large enough to make the first part of the Equation (4) higher than the maximum observed Hellinger distance; whereas the second factor is tied with the variations of these observed Hellinger values. These two factors are adjustable parameters, and can be properly tuned in the training period.

VI. Performance Evaluation

In this section, we evaluate the performance of vFDS in terms of detection accuracy and response time. Note that with the proper setting of threshold values, there will be no false alarm (i.e., false positive) under normal conditions. To measure false negatives, we use detection probability that is defined as the percentage of the successful identified attack instances over the total launched attacks in one set of experiments.

A. Detection of SYN Flooding Attacks

For `SYN` flooding attacks, our attacking traffic generation is based on the previous `SYN` flooding experiment [7], in which a proprietary SYN flooder program is used. The experiment findings show that a minimum of 500 `SYNs` per second is required to overwhelm the server. On the conservative side, our detection mechanism is designed to work with the lower bound of flooding attacks of 500 `SYNs` per second. We have used

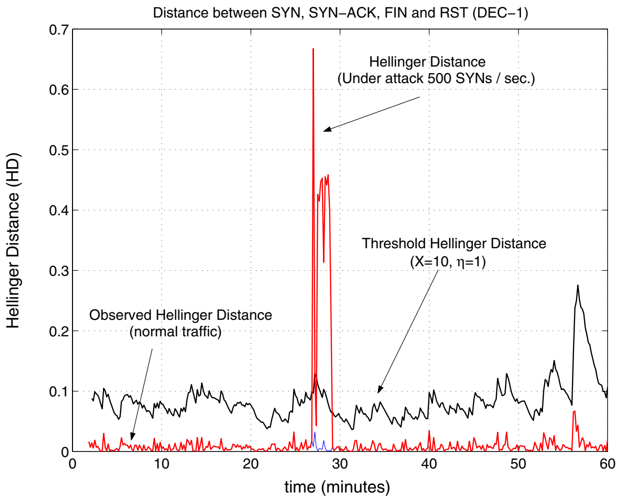


Fig. 8. Observed and Threshold Hellinger Distances (DEC-1)

DEC-1 trace as the normal background traffic (see Figure 2). The flooding traffic of various rate ($50-500$ SYNs per second) is mixed with the above normal background traffic. The flooding duration in all the experiments is assumed to be 30 seconds with the starting time randomly distributed between $10-55$ minutes. In our SYN flooding detection experiments, it has been empirically observed that the setting of $X = 10$ and $\eta = 1$ is good enough to capture all significant deviations in protocol attribute behaviors. The simulation results for different flooding rates are listed in Table I.

TABLE I

SYN FLOODING DETECTION PERFORMANCE OF vFDS

$(n = 10, \triangle t = 10 seconds, X = 10, \eta = 1)$

Flooding Rate	Detection Threshold	Number of Experiments	Detection Probability
50	$\simeq 0.09$	30	93.33%
75	$\simeq 0.09$	30	96.66%
100	$\simeq 0.09$	30	100%
500	$\simeq 0.09$	5	100%

Figure 8 shows the estimated threshold Hellinger distance (with $X = 10$ and $\eta = 1$) along with the observed Hellinger distance for DEC-1 trace. The measured distance of DEC-1 trace is always smaller than the estimated threshold. An alert flag will be raised only when the observed Hellinger distance of a particular testing period becomes higher than that of the estimated threshold Hellinger distance in that period. Figure 8 also shows, how the observed Hellinger distance dramatically changes with the introduction of SYN flooding traffic of 500 SYNs per second. The flooding traffic starts at 26.833 minutes and in the subsequent testing period of 27 minutes, the measured Hellinger distance shoots up to 0.668, easily crossing the threshold value and subsequently raising an alert flag.

B. Detection of SIP Flooding Attacks

In the following experiments, we defend a SIP proxy server against INVITE flooding attacks. Note that iSoftTech SIP Proxy Server [3] running on Linux PC (Pentium 3, 1GHz) and CISCO SIP Proxy Server [1], two popular commercial products, can handle 100 calls per second. Thus, there is no doubt that SIP proxy servers are susceptible to an INVITE flooding attack at the rate of 500 INVITEs per second.

The Hellinger distance plotted in Figure 7 shows the maximum observed distance of $8 * 10^{-3}$ and the average distance of $0.9 * 10^{-3}$. Therefore, by setting the threshold distance with $X = 20$ and $\eta = 1$, we can detect any significant deviation in the SIP traffic without raising a false alarm. In the INVITE flooding detection experiments, the SIP traffic generated in our testbed is used as the normal background traffic and is mixed with the flooding traffic varying from 50 to 500 INVITEs per second. The purpose of these experiments is to show that with the appropriate setting of the threshold value, vFDS not only identifies the flooding attack of 500 INVITEs per second with the accuracy of 100%, but also detects those flooding attacks with much lower flooding rates. The flooding duration of each experiment is set to 30 seconds, and the starting time of a flooding attack is randomly distributed between 10 to 55 minutes. The experimental results for different flooding rates are listed in Table II.

TABLE II

SIP INVITE FLOODING DETECTION PERFORMANCE OF vFDS

$(n = 10, \triangle t = 10 seconds, X = 20, \eta = 1)$

Flooding Rate	Detection Threshold	Number of Experiments	Detection Probability
50	$\simeq 0.02$	25	80%
75	$\simeq 0.02$	25	100%
100	$\simeq 0.02$	25	100%
500	$\simeq 0.02$	5	100%

Figure 9 illustrates the dynamics of the estimated threshold Hellinger distance and the observed Hellinger distance. Since the spikes of the observed Hellinger distance are much smaller than those of the estimated threshold distance, no false alarm is raised. The injection of attack traffic of 500 INVITEs per second at time 29.833 minutes causes the sudden surge of the observed Hellinger distance at next testing period, reaching to 0.3597. Because the observed Hellinger distance under the attack is much higher than the average threshold distance of 0.02, an alert flag is raised.

C. Detection of RTP Flooding Attacks

To detect RTP-based attacks, we generate the attacking traffic based on the experiment performed by Qovia Inc. [19]. They observed that SIP-based *Siemens Optiplus 600* phones with a G.711 codec performed well at 500 RTP PPS with a packet size of 200 bytes. However, as the RTP packet rate increases to 2500 PPS, the voice quality significantly deteriorates and subsequently the connection is broken. In our experiments, we assume UACs using the G.711 (i.e 50 PPS)

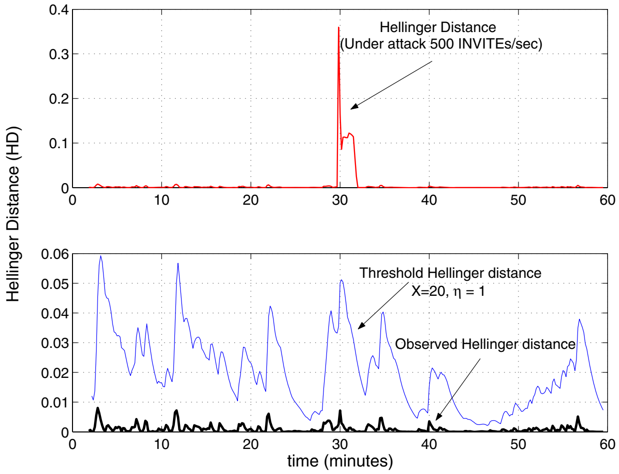


Fig. 9. Observed and Threshold Hellinger Distances (SIP)

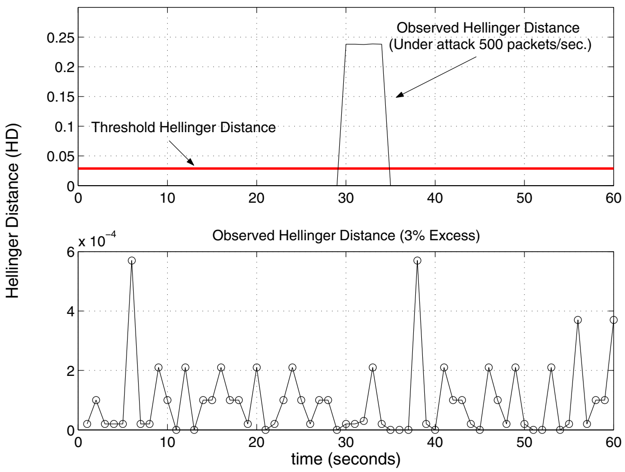


Fig. 10. Observed and Threshold Hellinger Distances (RTP)

codec algorithm. The RTP rate of 500 PPS or more indicates the onset of RTP flooding attacks.

The setting of a proper threshold distance for detecting RTP attacks is solely dependent on the RTP protocol attributes, without using the *stochastic gradient algorithm* applied in the previous two cases (i.e., the threshold settings for SYN and INVITE flooding attacks). The reason is primarily due to the deterministic nature of the RTP traffic compared to those of TCP and SIP.

Knowing the total number of incoming voice streams N and the media attribute $n^i_{\texttt{theoretical}}$ for each stream, we can compute the total number of expected incoming RTP packets as $\sum_{i=1}^{N} n^i_{\texttt{theoretical}}$. Any significant deviation from this expected value is an indication of an RTP attack. Now the questions are, how much deviation can be tolerated and how to set this tolerance limit as a threshold. To answer these two questions, first, for each $n^i_{\texttt{theoretical}}$ attribute, we introduce a new corresponding $n^i_{\texttt{threshold}}$ attribute that describes the upper-bound for the tolerable number of RTP packets per second. Thus, the tolerable upper-bound for the total number of RTP packets is $\sum_{i=1}^{N} n^i_{\texttt{threshold}}$. Second, we use the Hellinger distance method to quantify the difference between these two sums of attributes, and set the threshold as a static value based on the quantified result.

As an illustrative example, we consider a voice stream i with attributes $n^i_{theoretical}$ and $n^i_{threshold}$. The value of $n^i_{theoretical}$ is determined by the media encoding scheme used for the voice stream. To detect the flooding rate of 100 PPS, the value of $n^i_{threshold}$ is set to 100. For the G.711 codec algorithm, the $n^i_{theoretical}$ attribute is 50 PPS. Therefore, the threshold of Hellinger distance is computed as :

$$\begin{aligned} HD^i_{thresh.} &= (\sqrt{1/2} - \sqrt{q_{\texttt{theo.}}})^2 + (\sqrt{1/2} - \sqrt{q_{\texttt{thresh.}}})^2 \\ &= 0.029, \text{ where} \\ q_{\texttt{theo.}} &= n^i_{\texttt{theoretical}}/(n^i_{\texttt{theoretical}} + n^i_{\texttt{threshold}}) \\ q_{\texttt{thresh.}} &= n^i_{\texttt{threshold}}/(n^i_{\texttt{theoretical}} + n^i_{\texttt{threshold}}) \end{aligned}$$

Figure 10 plots the observed and threshold Hellinger distances for the voice stream. The flooding traffic of 500 PPS is injected into the voice stream at time 29 seconds and lasts for 5 seconds. As it is evident from the figure, the measured distances (with $\triangle t = 1$ second) of the RTP stream under flooding attack are ten times higher than the threshold distance, and hence, an alarm is raised.

D. Detection Time

Now we discuss how quickly an attack can be detected from its beginning. In the previous SYN and INVITE flooding detection experiments, 84% of them have the detection times between $13 - 18$ seconds, and for the rest of 16% of the experiments, their detection times are 10 seconds. In both set of experiments, our testing period (i.e. $\triangle t$) is fixed at 10 seconds. In the RTP flooding attacks, the testing period is one second. The observed attack detection delay is also $\simeq 1$ second. Overall, vFDS can quickly detect the various flooding attacks and the detection time varies between $1 - 2$ testing periods.

VII. Related Work

The previous works done by Wang et al. [27], [28] and Reynolds et al. [20] are the closest to our work. Wang et al. proposed a flooding detection system (FDS) based on the protocol behavior of TCP's control packet pairs. Reynolds et al. proposed Transport Layer Attack Sensor (TLAS) and Application Layer Attack Sensor (ALAS) to detect IP telephony flooding DoS attacks. At the SIP application layer, ALAS uses (INVITE, 200 OK) pair to detect IP telephony call request flood attacks. Although the (INVITE, 200 OK) pair is useful in detecting flooding attacks originated inside the enterprise network, its usage for detecting flooding attacks originated from the outside of the enterprise network is questionable. TLAS is based on TCP protocol behavior of (SYN, ACK) pairs. We do not use ACK packets in flooding detection, because it requires the state maintenance of TCP session and more processing power to distinguish these ACKs for control packets from those ACKs for data packets. Wu et al. [30] proposed SCIDIVE, a stateful cross protocol intrusion detection mechanism for VoIP. There are many other commercial network security products, which take the similar approach to validating the observed traffic behavior against the expected traffic behavior. *Mazu Profiler* [15] compares the current network activity with a baseline to detect suspicious behaviors. *Arbor Peakflow* [4] creates a baseline of network usage and detects anomalies. Instead of working on the aggregated traffic behaviors, these methods keep track of individual flows. However, maintaining states for each individual flow demands more memory and computational resources.

Hellinger distance is well studied in statistics and probability. Hellinger distance is a powerful tool for the study of product measures and pointwise differentiability in some asymptotic problems [18]. It is often used in machine learning and many other applications, such as regression, measuring ecological distances [29], viral email propagation [24], and data swapping [11].

VIII. Conclusions and Future Work

DoS attacks of flooding `SYN`, `INVITE` and RTP packets pose a serious threat to IP telephony infrastructure. The multi-protocol based VoIP service needs a fast and generic detection mechanism working across different protocol layers. This paper investigates the protocol attribute behaviors, and characterizes the network traffic with respect to the intrinsic correlation among protocol attributes. Utilizing Hellinger distance, we present an online statistical flooding detection mechanism, called vFDS, in which we measure the similarity (or dissimilarity) of the correlation among protocol attributes at different times. The rationale behind our approach is that a deviation from normal protocol behaviors can be measured and quantified. We exploit the extent of the deviation for detecting DoS attacks. We evaluate the effectiveness of vFDS using real Internet traces collected at the exchange points and the VoIP traces generated on the realistic SIP-based testbed. Our experimental results show that vFDS can achieve high detection accuracy with short detection time of $1-2$ testing periods. Our future work includes improving the detection sensitivity of vFDS against low-rate attacks that span longer period of time and conducting more realistic performance evaluation in the context of real VoIP traces.

References

[1] CISCO SIP Proxy Server. SIP High Availability Overview, www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/callc_c/sip_c/sipha_c/hachap1.htm, 2005.

[2] Ethereal. Network Protocol Analyzer, http://www.ethereal.com/, 2005.

[3] iSoftTech SIP Proxy Server. Software Design Overview Template, www.isofttech.com/downloads/SIP_3261_Proxy_Stack.pdf, 2005.

[4] Arbor Networks. Arbor Peakflow and Netflow. Product Overview, http:/www.arbornetworks.com/downloads/, 2006.

[5] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes : Theory and Applications*. Prentice Hall, 1st edition, 1993.

[6] M. Carson and D. Santay. *NIST Net Network Emulation Package*. Nist Net Web site : http://snad.ncsl.nist.gov/itg/nistnet/, June 1998.

[7] T. Darmohray and R. Oliver. Hot Spares for DoS attacks. The Magazine of USENIX and SAGE, ;login, 25(7), July 2000.

[8] DEC. Digital Equipment Corporation traces. Hourly Traffic traces, http://ita.ee.lbl.gov/html/contrib/DEC-PKT.html, 2005.

[9] A. Deslauriers, J. Pichitlamken, P. L'Ecuyer, and A. N. Avramidis. Markov chain models of a telephone call center with call blending. Technical report, GERAD and DIRO, University of Montreal, 2003. Preprint.

[10] M. Fannes and P. Spincemaille. The mutual affinity of random measures. In *eprint arXiv:math-ph/0112034*, December 2001.

[11] S. Gomatam, A. F. Karr, C. C. Liu, and A. P. Sanil. Data swapping: A risk-utility framework and web service implementation. In *DG.O*, 2003.

[12] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, IETF Network Working Group, 1998.

[13] V. Jacobson and M. J. Karels. Congestion avoidance and control. In *Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, volume 18, 4, pages 314–329, 1988.

[14] G. Jongbloed and G. Koole. Managing uncertainty in call centers using Poisson mixtures. *Applied Stochastic Models in Business and Industry*, 17:307–318, 2001.

[15] Mazu Networks. Mazu Profiler. Product Overview, http:/www.mazunetworks.com/resources/product-sheets/, 2006.

[16] NLANR, Front Range GigaPOP. NLANR network traffic traces. Daily traffic traces, http://pma.nlanr.net/Traces/, 2005.

[17] Paul Russell. netfilter/iptables. Firewall, http://www.netfilter.org/, 2005.

[18] D. Pollard. *Asymptopia*. http://www.stat.yale.edu/ pollard/ (Book in Progress), 1st edition, 2000.

[19] Qovia Inc. Network Intrusion and QoS impact in VoIP. White paper, http://www.qovia.com/, August 2004.

[20] B. Reynolds and D. Ghosal. Secure IP Telephony using Multi-layered Protection. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, February 2003.

[21] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, IETF Network Working Group, 2002.

[22] H. Sengar, D. Wijesekera, H. Wang, and S. Jajodia. VoIP Intrusion Detection Through Interacting Protocol State Machines. In *IEEE Dependable Systems and Networks Conference (DSN 2006)*, June 2006.

[23] W. Stevens. *TCP/IP Illustrated Volume-1*. Addison-Wesley Publishing Company, 1st edition, 1994.

[24] S. J. Stolfo, W.-J. Li, S. Hershkop, K. Wang, C.-W. Hu, and O. Nimeskern. Detecting viral propagations using email behavior profiles. *ACM Transactions on Internet Technology (TOIT)*, May 2004.

[25] Telecost. Telecost : On Call Durations. Product Overview, http://www.telecost.co.uk/Products/OnCallDurations.htm, 2005.

[26] K. Thompson, G. J. Miller, and R. Wilder. Wide-Area Internet Traffic Patterns and Characteristics. In *IEEE Network*, volume 11, November/December 1997.

[27] H. Wang, D. Zhang, and K. G. Shin. Detecting syn flooding attacks. In *IEEE INFOCOM*, June 2002.

[28] H. Wang, D. Zhang, and K. G. Shin. Syn-dog: Sniffing syn flooding sources. In *IEEE ICDCS*, July 2002.

[29] World Agroforestry Center. Regression and Analysis of Variance. Tutorial, http://www.worldagroforestry.org/, 2005.

[30] Y. Wu, S. Bagchi, S. Garg, N. Singh, and T. Tsai. SCIDIVE: A Stateful and Cross Protocol Intrusion Detection Architecture for Voice-over-IP Environments. In *IEEE Dependable Systems and Networks Conference (DSN 2004)*, June 2004.