Detecting VoIP Floods Using the Hellinger Distance

Hemant Sengar, *Student Member*, *IEEE*, Haining Wang, *Member*, *IEEE*, Duminda Wijesekera, *Senior Member*, *IEEE*, and Sushil Jajodia, *Senior Member*, *IEEE*

Abstract—Voice over IP (VoIP), also known as Internet telephony, is gaining market share rapidly and now competes favorably as one of the visible applications of the Internet. Nevertheless, being an application running over the TCP/IP suite, it is susceptible to flooding attacks. If flooded, as a time-sensitive service, VoIP may show noticeable service degradation and even encounter sudden service disruptions. Because multiple protocols are involved in a VoIP service and most of them are susceptible to flooding, an effective solution must be able to detect and overcome hybrid floods. As a solution, we offer the *VoIP Flooding Detection System (vFDS)*—an online statistical anomaly detection framework that generates alerts based on abnormal variations in a selected hybrid collection of traffic flows. It does so by viewing collections of related packet streams as evolving probability distributions and measuring abnormal variations in their relationships based on the *Hellinger distance*—a measure of variability between two probability distributions. Experimental results show that vFDS is fast and accurate in detecting flooding attacks, without noticeably increasing call setup times or introducing jitter into the voice streams.

Index Terms—VoIP, flooding attacks, Hellinger distance.

1 Introduction

IP telephony, commonly known as *Voice over IP (VoIP)*, provides a viable alternative to Public Switched Telephones (PSTNs). As its deployment spreads, VoIP is likely to become a prime target of attacks, of which flooding lists high, perhaps due to its simplicity and the abundance of tool support. Since VoIP is a time-sensitive service, flooding can easily deteriorate the perceived quality of voice services (QoS), and even cripple down the devices in the path from caller to callee, such as IP telephones, SIP proxy servers, and softswitches.

Unlike the majority of Internet services, nonproprietary VoIP services use many protocols to control calls and deliver audio streams, such as the *Session Initiation Protocol (SIP)* [26] for call setup and teardown and the *Real-time Transport Protocol (RTP)* [28] to deliver voice packets, etc. Because packet floods can be generated for any combination of protocols used for VoIP, a defense mechanism that can detect hybrid (or polymorphic) packet floods is desired. Also, due to the time-sensitive nature of the application, such a mechanism must not introduce noticeable timing delays to transaction-like control flow or jitters to audio streams.

As a solution, we propose a statistical abnormal behavior detection mechanism called *VoIP Flooding Detection System (vFDS)*, which is suitable for detecting hybrid packet floods. vFDS is based on the simple observation that despite the burstiness of packet-level Internet traffic (including VoIP traffic), transaction-like control traffic generated by the VoIP protocol suite still maintains the inherent attribute correlations. In general, vFDS learns and quantitatively tracks such relationships among chosen attributes of VoIP packet streams, and raises an alarm for observed significant deviations, which alert an onset of a flooding attack.

In order to quantify the correlations between chosen attributes, vFDS views such packet streams as unfolding data belonging to some sample space of a probability distribution and uses the *Hellinger distance (HD)* [23]—a metric that quantifies the deviation between two probability measures. Compared to other distance measures, the selection of HD is due to the following three reasons: 1) it is not computationally intensive, 2) it does have a natural lower and upper bounds of 0 and 1, respectively, and 3) it is based on the proportion of the protocol attributes. We validate the effectiveness of vFDS using Internet and VoIP traffic traces. Our experimental results demonstrate that vFDS can achieve high detection accuracy with a short detection time while introducing no perceivable delay to call setup times and perceptible jitter to audio streams, utilized by a host of SIP-controlled VoIP protocols.

Previous protocol-behavior-based solutions [25], [35], [36] are limited in their capabilities, mainly because they have been custom-crafted for a specific protocol and accordingly track selected protocol attribute pairs specific to that protocol. For example, Reynolds and Ghosal's [25] ALAS and TLAS used the difference between {INVITE, 200 OK} and {SYN, ACK} attribute pairs for detecting INVITE and SYN flooding attacks, respectively (that is, request for opening connections in SIP and TCP that do not complete). Wang et al. [35] used the {SYN, FIN} pair for

- *H. Sengar, D. Wijesekera, and S. Jajodia are with the Center for Secure Information Systems, George Mason University, 4400 University Drive, Fairfax, VA 22030. E-mail: {hsengar, dwijesek, jajodia}@gmu.edu.*
- *H. Wang is with the Department of Computer Science, College of William and Mary, PO Box 8795, Williamsburg, VA 23187. E-mail: hnw@cs.wm.edu.*

Manuscript received 10 May 2007; revised 31 Aug. 2007; accepted 18 Sept. 2007; published online 27 Sept. 2007.
Recommended for acceptance by A. Boukerche.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2007-05-0149.
Digital Object Identifier no. 10.1109/TPDS.2007.70786.

SYN flooding attack detection. These solutions—being custom crafted for each individual pair of packets—are not sufficiently generic for detecting mixed traffic floods with many such pairs such as ({INVITE, 200 OK}) and ({SYN, FIN}). Consequently, they cannot detect polymorphic flooding attacks launched with hybrid traffic streams.

Another potential detection mechanism, namely, benchmarking correct protocol behavior on IP telephony, does not work either, because many observed RTP-based voice streams do not show patterns at the packet level. Moreover, research shows that modeling phone call arrivals as a random process with a deterministic time-varying arrival rate is not possible [9], [18].

The remainder of this paper is structured as follows: Section 2 briefly describes the background of this study. Section 3 presents the threat model. Section 4 describes the system design of vFDS. Section 5 profiles normal protocol behaviors and discusses the inherent correlation among protocol attributes. Section 6 shows how to compute HD for different traffic streams. Section 7 evaluates the performance of vFDS. Section 8 discusses the impact of vFDS upon the quality of VoIP services. Section 9 surveys related work, and Section 10 concludes the paper.

2 Background

2.1 SIP-Based IP Telephony

SIP [26], running as the standard signaling protocol for VoIP at the application layer, is used to set up, modify, and tear down multimedia sessions between two or more participants. Sometimes referred to as the SS7 *of future telephony* [17], it transmits the message body in clear text. SIP call control uses the *Session Description Protocol (SDP)* [14] for describing multimedia sessions.

2.1.1 SIP Architectural Components

SIP-based telecommunication architectures have two kinds of elements: end devices, referred to as *user agents (UAs)*, and *SIP servers*. Irrespective of being a software or hardware phones, UAs combine two subentities: the connection requester referred as the *UA client (UAC)* and the connection request receiver referred to as the *UA server (UAS)*. Consequently, during a SIP session, both UAs switches back and forth between the UAC and UAS functionalities. RFC 3261 [26] describes four types of SIP-implementation-dependent logical servers: *Location Servers*, *Redirect Servers*, *Registrar Severs*, and *Proxy Servers*.

2.1.2 SIP Messages and Operations

Influenced by two widely used Internet protocols, namely, the *Hypertext Transfer Protocol (HTTP)* [12] and the *Simple Mail Transfer Protocol (SMTP)* [19], SIP messages consisting of request-response pairs are exchanged for a call setup. The SIP request are also called methods, and there are six of them (INVITE, ACK, BYE, CANCEL, REGISTER, and OPTIONS) described in [26]. 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.

Fig. 1 shows a typical message flow for a call setup between UAC UA-A and UAS UA-B. Assuming that the two UAs belong to different domains with their own proxy servers, UA-A's proxy server uses its Domain Name Service to locate a proxy server for UA-B. After obtaining the IP address of UA-B's proxy server, UA-A's proxy server sends an INVITE request to the latter with UA-B's name. In response, UA-B's proxy server consults a location service database to find out the current location of UA-B and forwards the INVITE request to the UA-server residing on UA-B's SIP phone. Exchanging INVITE, 200 OK, and ACK messages completes the three-way handshake and establishes a SIP session. Then, an SDP compliant set of parameters is exchanged in SIP message bodies and finally establishes an RTP stream to exchange audio data.

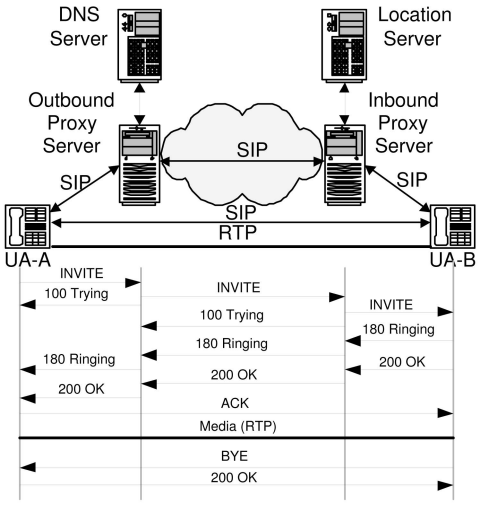


Fig. 1. SIP call setup example.

SIP proxy servers have no media capabilities and only facilitate the two end points (that is, IP telephones) to discover and contact each other through SIP signaling. Once the end points have been located, the media flows directly between UAs without going through proxies using a path independent of the one used by SIP signals.

At the end of the call, one party hangs up, resulting in that party's agent sending a BYE message to terminate the session and receiving a 200 OK response from its counterpart. This example shows the basic functionality of SIP, described in more detail in RFC 3261 [26].

2.2 Placing the vFDS in an Enterprise IP Telephone Network

Current enterprise VoIP networks consist of a network interior and a demilitarized zone (DMZ), as shown in Fig. 2. The DMZ may contain many servers, including a SIP proxy server. Under the assumption that most VoIP attacks come from outside the enterprise network, because of the need to inspect all VoIP traffic flowing through the enterprise (including the DMZ), vFDS is strategically located between the edge router and the firewall, as shown in Fig. 2. This placement of vFDS also obviates the need for flood detection mechanisms at individual SIP entities. In a practical deployment, vFDS can be transparently interposed at either an edge router or a firewall and be implemented as a loadable module of the router or the firewall.

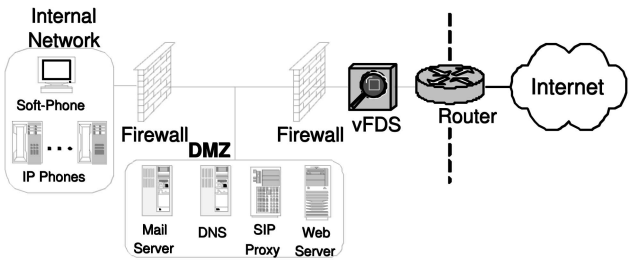


Fig. 2. Enterprise IP telephony network.

Note that vFDS is complementary to the existing VoIP cryptographic security mechanisms. Since vFDS is in the proximity to the SIP proxy server, the inspection of the SIP header should not be a problem. Moreover, although the voice stream is encrypted from end to end, we only count the number of RTP packets and do not examine its contents.

3 The Threat Model

As stated, an enterprise VoIP service may receive hybrid packet floods for many protocols, but our analysis is focused on `(SYN)-`, `(INVITE)-`, and `RTP`-related floods, belonging to transport and applications layers carrying call control and audio packets.

3.1 Transport-Layer Floods

The transport layer carries SIP signals over TCP, SCTP, or UDP, while RTP-based media streams over UDP. Consequently, the effect on SIP-based VoIP due to transport-level floods is indirect. Although both TCP and UDP are susceptible to flooding attacks, being stateless, UDP is less vulnerable to floods than TCP. However, SIP's reliability mechanism used in UDP transportation makes it susceptible to UDP floods too.

At the transport layer, there are numerous protection devices from many different vendors. Most of these protection mechanisms are based on rate-limiting solutions and maintaining the flow-level (source-destination) relationship. These data-centric solutions are prone to DoS attacks and are not scalable to voice carrier networks. It is important to note that between the source (caller) and the destination (callee), only few signaling messages are exchanged. Compared with a brute-force UDP flooding attack, it is much easier for an attacker to succeed a SIP flooding attack with much less effort (that is, by generating much less flooding SIP traffic to knock down a SIP proxy). That is the major reason why we still need SIP-level detection at the application layer.

3.2 Application-Layer Floods

As mentioned, SIP and RTP are the application-level protocols we consider. SIP entities are susceptible to two kinds of flooding attacks: The first directs bogus traffic to exhaust resources, and the second exploits protocol vulnerabilities. Being transactional due to having (request, response) pairs, SIP agents have to maintain a transactional state until the transaction completes or the receiving agent times out. For example, a SIP proxy can optionally maintain an `INVITE` transaction state up to 3 minutes [26]. Similarly,

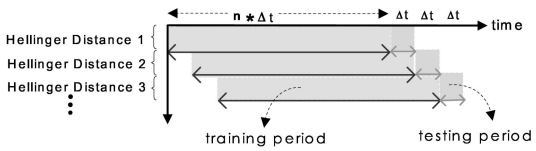


Fig. 3. Relationship between training and testing periods.

when a UAS conditionally accepts an `INVITE` request, it can generate an `2XX` response and wait for an `ACK` while maintaining its state. Additionally, unlike PSTN, IP phones can generate simultaneous multiple call requests, such as four using the `ZIP` 4×4 phone by Zultys Technologies [39] and nine using the `Alti-IP 600` phone by AltiGen Communications [3], making it easy to create `INVITE` request floods with few phones. Thus, stateful proxies and UASs can be easily flooded.

RTP delivers live media streams between callers and callees. An RTP flooding attack exploits the vulnerabilities of the media path to deteriorate the perceived voice quality. This attack is created by sending a barrage of fabricated RTP packets without following any media encoding scheme, with the objective of exhausting the available bandwidth and sometimes making IP phones dysfunctional.

4 The vFDS Design

In general, vFDS detects anomalies in collections of packet streams, going through a cyclic behavior consisting of two phases: the training and testing phases. As shown in Fig. 3, during the training phase, the training data set consisting of the attribute set is collected over n sampling periods of duration $\triangle t$ over a normal traffic stream. This initial training data set is assumed to be devoid of any attacks and acts as a base for comparing with the next $(n+1)$th periods of the testing data set. Using the soon-to-be-described HD, we measure the distance between these two data sets. If the measured distance exceeds a threshold, an alarm is raised; otherwise, the testing data set is included in the immediately preceding $(n-1)$ 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 order for this design to work, the following three parameters are computed online:

1. *The probabilistic distribution for training data.* This is computed as the ratio of packets that satisfy the feature to the total number of packets received during the training phase. Section 5 describes the details of how these are computed for TCP, SIP, and RTP.
2. *The probabilistic distribution for testing data.* This is computed as averages during the time window immediately following the training period, again as a ratio of packets satisfying the chosen feature to the total number of packets, whereas the deviation of the two probability distributions are computed using the (soon-to-be-described) HD.

3. *The threshold of deviation to distinguish normal behavior from the abnormal behavior.* This is used to compute a dynamic threshold as the computation progresses through cycles of training and testing phases, using Jacobson's fast deviation computing algorithm [16].

4.1 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* [23] but with several advantages. To explain this, let $\mathbb{P}$ and $\mathbb{Q}$ be two probability distributions on a finite sample space Ω, where $\mathbb{P}$ and $\mathbb{Q}$ on Ω are N-tuples $(p_1, p_2, \ldots, p_N)$ and $(q_1, q_2, \ldots, q_N)$, respectively, satisfying (in)equalities $p_\alpha \geq 0$, $q_\alpha \geq 0$, $\sum_\alpha p_\alpha = 1$, and $\sum_\alpha q_\alpha = 1$. Then, the HD 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 HD satisfies the inequality $0 \leq d_H^2 \leq 1$, and $d_H^2 = 0$ when $\mathbb{P} = \mathbb{Q}$. Disjoint $\mathbb{P}$ and $\mathbb{Q}$ shows the maximum distance of one. Sometimes, the factor $\frac{1}{2}$ is not used in the above equation. A related notion is the *affinity* between probability measures, which is defined as

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

The affinity between two probability measures $\mathbb{P}$ and $\mathbb{Q}$ is one (that is, $A = 1$) if they are equal and zero if the measures are *totally different*. Further details on HD can be found in [23] and [11].

4.1.1 Measuring Protocol Deviations Using the Hellinger Distance

In order to detect protocol violations, depending upon the protocol to be observed and a collection of potential attacks that can launched against it, we select and track the distribution of a (small) set of attributes. Suppose we choose N attributes of a protocol, which satisfy p_α, $q_\alpha \geq 0$, $\sum_\alpha p_\alpha = 1$, and $\sum_\alpha q_\alpha = 1$. 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 the normalized frequencies of all N attributes.

4.2 Detection Threshold

Normal attribute behaviors also change with time, although the strong attribute correlation makes the fluctuation of its dynamics much less than that of traffic behaviors. To accurately keep track of the normal attribute behaviors, we use a dynamic threshold for detection. Such a dynamic setting of threshold will make an attack harder to evade. We employ the *stochastic gradient algorithm* to compute the *dynamic threshold* based on the HD observed during the previous training period. Our threshold is an instance of Jacobson's *Fast algorithm for RTT mean and variation* [16]. Fast estimators for average a and mean deviation ν, given measurement m, can be computed as

$$Err = m_n - a_{n-1}, \tag{1}$$

$$a_n \longleftarrow a_{n-1} + g.Err, \tag{2}$$

$$\nu_n \longleftarrow \nu_{n-1} + h.(|Err| - \nu_{n-1}), \tag{3}$$

where m_n is the current sample of the HD, a_{n-1} and a_n are the previous and current smoothened Hellinger distances, respectively, and 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 negative exponents of two. Here, we use the values $g = \frac{1}{2^3}$ and $h = \frac{1}{2^2}$, as previous research suggested [31, Chapter 21]. Although the original g and h are used in the context of RTT measurement, the underlying principles of both cases are the same: based on the past and present values, we attempt to predict the future values. The smoothened HD a_n is based on the observed HD m, which is measured between the probability measures $\mathbb{P}$ and $\mathbb{Q}$. During the testing periods, we derive the estimated threshold HD ($HD^{thresh.}$) using the smoothened HD (2) and the mean deviation (3):

$$HD_{n+1}^{thresh.} = \underbrace{X * a_n}_{} + \underbrace{\eta * \nu_n}_{}. \tag{4}$$

The purpose of the multiplication factors X and η is to get 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 (4), which largely depends upon the observed HDs, should be large enough to make the first part of (4) higher than the maximum observed HD, 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 during the training period.

5 Profiling Normal Protocol Behaviors

We use real Internet traces and the VoIP traces obtained from our testbed to experimentally profile normal protocol behaviors.

5.1 Benchmarking TCP Behavior

To study the TCP attribute behaviors, we choose two sets of traces representing real-life Internet traffic at the exchange points that connect stub networks to the Internet. The collection times of the two sets of traces are deliberately chosen to be 10 years apart, to demonstrate the invariant nature of the TCP attribute behaviors, irrespective of changing Internet traffic. The first set of traces (with bidirectional traffic) were gathered from the *Front Range GigaPOP* (FRGP) [21], where one trace (FRGP-1) was originally collected on Saturday, October 1, 2005, and the other (bidirectional) trace (FRGP-2) was collected on Tuesday, November 1, 2005. We intend to have FRGP-1 and FRGP-2 to be one month apart to further demonstrate the existence of the inherent correlation among protocol attributes. The second set of trace is the collection of 1 hour's worth of Wide Area Network (WAN) traffic between the *Digital Equipment Corporation* (DEC) [8] and the

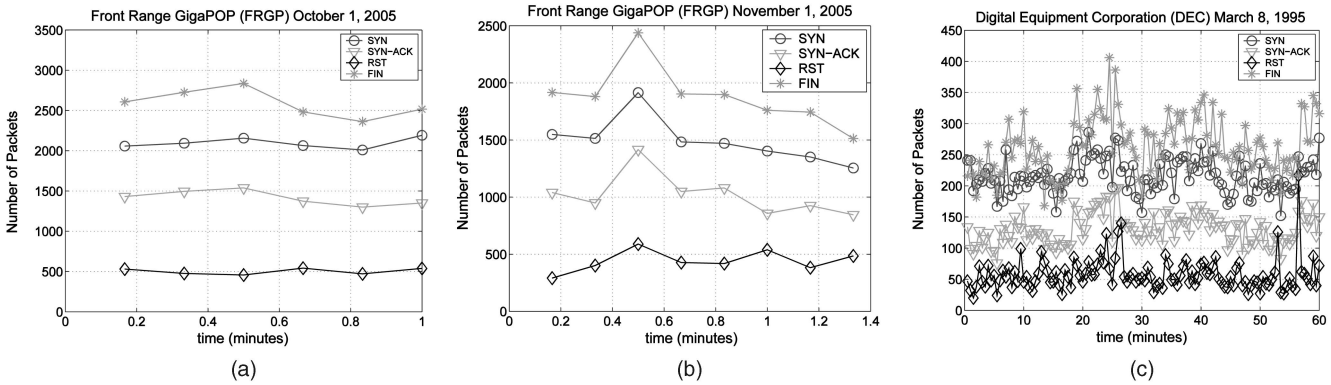


Fig. 4. TCP attribute behaviors in various traces. (a) FRGP-1. (b) FRGP-2. (c) DEC.

rest of the Internet. The trace ran from 22:00 to 23:00 on Wednesday, March 8, 1995. Both these traces are bidirectional. We parse the traces and extract SYN, SYN-ACK, FIN, and RST packets from the TCP streams.

Fig. 4 illustrates the behaviors of the SYN, SYN-ACK, FIN, and RST attributes of the TCP streams in FRGP-1, FRGP-2, and DEC, respectively. In the normal TCP handshake process, for each SYN request from the client, there is one SYN-ACK response from the server. However, Fig. 4, does not show an exact one-to-one mapping between SYN and SYN-ACK. The curve of SYN is clearly above that of SYN-ACK. Some plausible reasons for this observed discrepancy are SYN losses and consequent retransmissions thereof, perhaps due to a dead or heavily overloaded server that does not generate SYN-ACKs in return. Also, under normal conditions, a TCP connection starts with a SYN packet and is torn down by the two exchanging FIN packets, due to the back and forth exchange between the client and the server during connection termination. However, our observations in Fig. 4 shows that the FIN curve lies above the SYN curve but not always at twice the height of the SYN-ACK curve. This discrepancy can be attributed to the fact that not every observed SYN-ACK leads to an established TCP connection, and also, an RST packet can terminate an established TCP connection without generating any FIN packets.

5.2 Benchmarking SIP Behavior

In order to study the attribute behaviors of VoIP traffic, we build a testbed including SIP proxy servers and SIP-based soft-phones. The testbed consists of four PCs (500-MHz Pentium III CPU with 128 Mbytes of RAM) equipped with Linux operating system acting as SIP clients, SIP servers, and routers. Fig. 5 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 playing the role of multiple UACs attempting to call UASs in the enterprise network C.

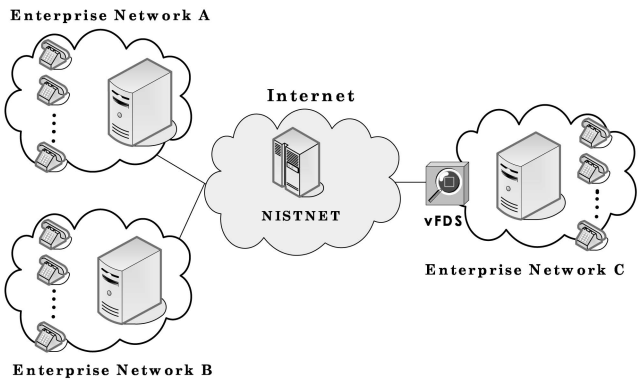


Fig. 5. Layout of the SIP-based VoIP testbed.

The average call generation rate is 50 calls per second, with the lowest call rate being 25 calls per second and the peak call rate being 70 calls per second. The *talking time* is set to 60 seconds. The voice codec algorithm used is *G.711* (50 packets per second (PPS)). The WAN emulator *NISTNet* [5] connects networks A and B to network C using multiple 100-Mbps Ethernet links. NISTNet runs on a Linux router where the 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 percent in our experiments. We use the Network Time Protocol (NTP) to synchronize the time of clients with that of the NISTNET server. SIP signaling messages are carried by UDP. SIP timer $T1$ plays a significant role in packet retransmission. We set $T1$ to 500 ms, which is recommended by the SIP standard considering the normal end-to-end delay over the Internet. The experimental run lasted an hour.

Fig. 6 plots the observed SIP attribute behaviors at the SIP proxy server of the enterprise network C. As shown in Fig. 6, the 200 OK[1] and ACK curves closely overlap with each other, whereas there are occasional small gaps between them. During the 1-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, strong positive correlations between INVITE, 200 OK (in three-way handshake), ACK, and BYE messages are shown in Fig. 6.

5.3 Benchmarking RTP Behavior

Compared to TCP and SIP, RTP does not have an inherent transactional behavior to observe in the form of message or packet pairs. At the application layer, we can only observe the

1. Here, the 200 OK messages are those in the three-way handshake phase only.

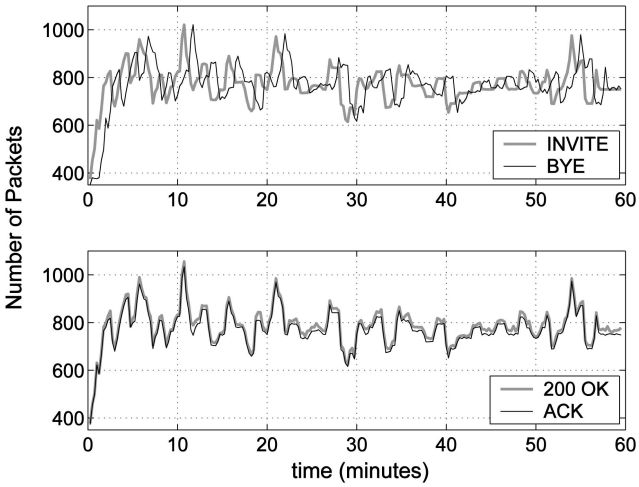


Fig. 6. SIP attribute behaviors.

number of RTP packets received per time unit. Based on those RTP packets passing by, 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 $n_{observed}$ (that is, the observed number of packets in a real RTP stream), where both attributes represent the number of packets in a given time interval. At the application layer, vFDS computes the value of $n_{observed}$ by counting the number of incoming RTP packets for each voice stream that is identified by the destination (IP address, port number) combination. To determine the value of $n_{theoretical}$, we need to incorporate the communication between the SIP and RTP state machines [30]. Because it monitors all packets for each call, vFDS can fulfill this requirement. The call control (SIP) and media delivery (RTP) protocols are synchronized by exchanging the synchronization messages for critical events in the established sessions. Media attributes such as the format, the encoding algorithm, and the sampling rate that are included in the SIP message body are accessible to the RTP state machine via the SIP state machine. Note that the renegotiations through INVITE messages are also taken into account. Suppose that the media information for a caller is

$$\langle \text{v.media\_format} = \text{audio},\ \text{v.media\_encoding} = \text{PCMU}\ (\text{that is G.711}),\ \text{v.sampling\_rate} = 8{,}000\rangle.$$

Then, the number of packets per second 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 the $n_{theoretical}$ 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 as shown in the following equation, instead of keeping track of the $n_{theoretical}$ attribute for each caller's RTP stream individually:

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

In the equation above, there are $n^{i}_{theoretical}$ packets in virtual stream i among N total number of open virtual

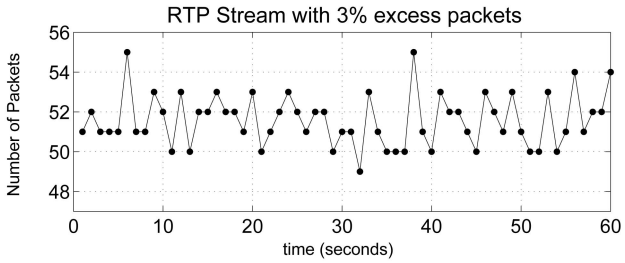


Fig. 7. RTP attribute behaviors.

streams during the $\triangle t$ time period. That models N active calls, each with its own incoming RTP stream and each stream with its own negotiated media encoding scheme.

In order to observe the RTP attribute distribution, we assume that UACs use G.711 (that is, $n_{theoretical} = 50$ PPS) codec algorithm. Fig. 7 shows one instance of the simulated RTP stream trace with 3 percent duplicate (that is, excess) packets. In this example, we have considered only one incoming media RTP stream, but it could be generalized to include any number of RTP streams.

5.4 Inherent Attribute Correlation

As seen, collected traces do not show the ideal one-to-one mapping between protocol attributes. 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 (that is, intrinsic correlation) do not vary much with the change of time and have an observable correlation with the total number of packets.

6 Computing Hellinger Distances

We analyze traffic and classify packets, first, at the transport layer and, then, at the application layer. The data sampling duration at both protocol layers are set to $\triangle t$, which determines both the detection resolution of flooding attacks and the computational overhead of vFDS.

6.1 Data Sampling

Most TCP connections last for 12 to 19 seconds [34], whereas IP phone calls last much longer: 50 percent of calls complete around 1 minute, and 10 percent of calls last even longer than 10 minutes [33]. Consequently, 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 1 minute, respectively. Fortunately, our detection mechanism is not sensitive to individual per-flow state information and is based only on the correlation between 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 $\triangle t$, the HD also depends upon the training period $(n * \triangle t)$. A longer training period provides a more accurate distance, whereas a shorter training period adapts

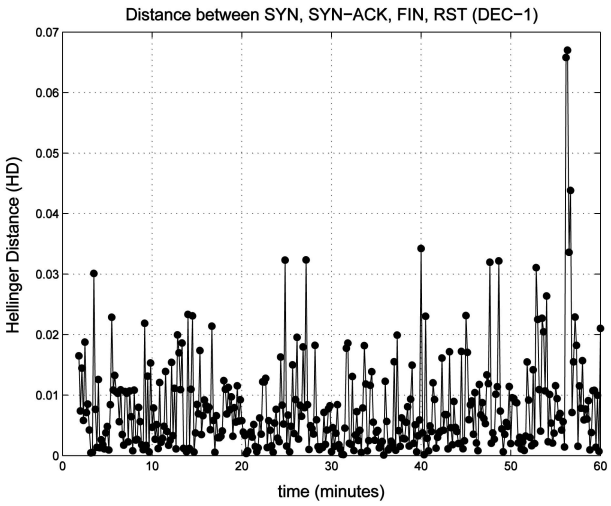


Fig. 8. HD of TCP attributes.

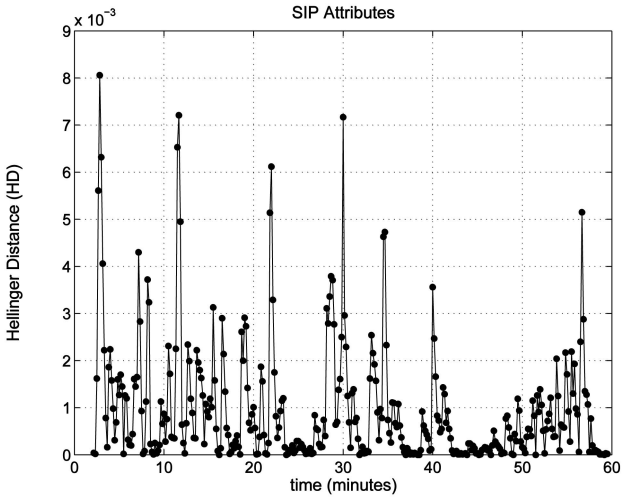


Fig. 9. HD of SIP attributes.

quicker to the changing dynamics of network traffic. To balance the accuracy and responsiveness, we set the training period to 120 seconds (that is, $n = 12$ samples) in all of the traces and obtained the results in Fig. 8.

6.2 Computing the Hellinger Distance for TCP

In this experiment, we choose four attributes SYN, SYN-ACK, FIN, and $\texttt{RST}_{active}$ belonging to TCP packets and apply the threshold filter in [35] to filter out $\texttt{RST}_{passive}$ from observed RSTs. Henceforth, we do not distinguish between RST and $\texttt{RST}_{active}$ packets.

Now, suppose that there are N_{SYN}, $N_{\text{SYN-ACK}}$, N_{FIN}, and N_{RST} packets during the training period (that is, in $n * \triangle t$ time). $\mathbb{P}$ is an array of the normalized frequencies of p_{SYN}, $p_{\text{SYN-ACK}}$, p_{FIN}, and p_{RST} over the training data set, and $\mathbb{Q}$ is an array of the normalized frequencies of q_{SYN}, $q_{\text{SYN-ACK}}$, q_{FIN}, and q_{RST} of the same attributes observed over the testing period (that is, at the $(n+1)$th sampling duration), defined as follows:

$$\begin{aligned} p_\alpha &= N_\alpha / N_{Total}, \\ &\quad \text{where } \alpha \in \{\texttt{SYN}, \texttt{SYN}-\texttt{ACK}, \texttt{FIN}, \texttt{RST}\}, \text{ and} \\ &\quad N_{Total} = (N_{\text{SYN}} + N_{\text{SYN-ACK}} + N_{\text{FIN}} + N_{\text{RST}}), \\ q_\alpha &= N'_\alpha / N'_{Total}, \\ &\quad \text{where } \alpha \in \{\texttt{SYN}, \texttt{SYN}-\texttt{ACK}, \texttt{FIN}, \texttt{RST}\} \text{ and} \\ &\quad N'_{Total} = (N'_{\text{SYN}} + N'_{\text{SYN-ACK}} + N'_{\text{FIN}} + N'_{\text{RST}}). \end{aligned}$$

The HD between $\mathbb{P}$ and $\mathbb{Q}$ at the end of $(n+1)$th sampling period is computed as follows:

$$\begin{aligned} HD_1 = &(\sqrt{p_{\text{SYN}}} - \sqrt{q_{\text{SYN}}})^2 + (\sqrt{p_{\text{SYN-ACK}}} - \sqrt{q_{\text{SYN-ACK}}})^2 \\ &+ (\sqrt{p_{\text{FIN}}} - \sqrt{q_{\text{FIN}}})^2 + (\sqrt{p_{\text{RST}}} - \sqrt{q_{\text{RST}}})^2. \end{aligned}$$

Fig. 8 shows the plot of HD for the DEC trace taking all four attributes at the same time. Throughout the 1-hour duration, the TCP attribute behavior of the DEC trace shows a remarkable similarity with time, given the fact that an HD of zero (that is, HD $= 0.0$) represents the same probability measures. The DEC trace sample has an average low distance of 0.007 and a maximum distance of 0.064.

6.3 Computing the Hellinger Distance for SIP

We choose to experiment with SIP attributes INVITE, 200 OK, ACK, and BYE. Here, the probability measure $\mathbb{P}$ is an array of the normalized frequencies of p_{INVITE}, $p_{200\text{ OK}}$, p_{ACK}, and p_{BYE} over the training data set. Similarly, $\mathbb{Q}$ is an array of q_{INVITE}, $q_{200\text{ OK}}$, q_{ACK}, and q_{BYE} during the chosen testing period. All other details are similar to the previous example. To calculate the HD between $\mathbb{P}$ and $\mathbb{Q}$, we use

$$\begin{aligned} HD = &(\sqrt{p_{\text{INVITE}}} - \sqrt{q_{\text{INVITE}}})^2 + (\sqrt{p_{200\text{ OK}}} - \sqrt{q_{200\text{ OK}}})^2 \\ &+ (\sqrt{p_{\text{ACK}}} - \sqrt{q_{\text{ACK}}})^2 + (\sqrt{p_{\text{BYE}}} - \sqrt{q_{\text{BYE}}})^2. \end{aligned}$$

Fig. 9 shows the HD for the SIP attribute set of {INVITE, 200 OK, ACK, 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 HD indicates the closeness between the training and observed traffic behaviors.

6.4 Computing the Hellinger Distance for RTP

In this experiment, we choose RTP and its derived attributes $n_{theoretical}$ and $n_{observed}$. The probability measure $\mathbb{P}$ at time $t = 0$ is

$$\begin{aligned} p_{theo.} &= n_{theoretical}/(n_{theoretical} + n_{observed}), \\ p_{obs.} &= n_{observed}/(n_{theoretical} + n_{observed}), \end{aligned}$$

where both $n_{theoretical}$ and $n_{observed}$ are initialized to 50 PPS, thus giving the values of $p_{theo.} = p_{obs.} = 1/2$. $\mathbb{P}$ remains constant for the subsequent sampling periods, except when it is changed by a SIP re-INVITE message (that is, change of media encoding scheme). The $\mathbb{Q}$ for each testing period $\triangle t$ is calculated as

$$\begin{aligned} q_{theo.} &= n_{theoretical} * \triangle t/(n_{theoretical} * \triangle t + n_{observed}), \\ q_{obs.} &= n_{observed}/(n_{theoretical} * \triangle t + n_{observed}). \end{aligned}$$

$n_{observed}$ is the actual number of RTP packets observed for a particular voice stream during the $\triangle t$ time period. At the

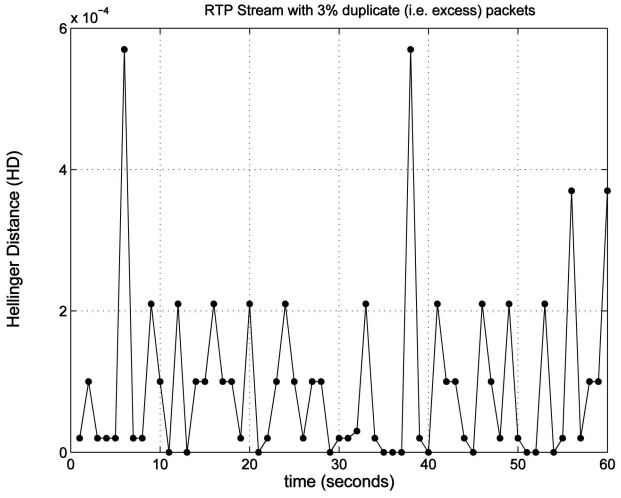


Fig. 10. HD of RTP attributes.

end of the first sampling period, the HD for the caller's media stream is computed as

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

For the subsequent testing periods, the HD is computed by changing only the values of $q_{theo.}$ and $q_{obs.}$ for that particular $\triangle t$. Fig. 10 shows the observed HDs for an RTP stream trace with 3 percent duplicate packets, showing observed distances in the order of $\simeq 10^{-4}$.

7 Detection Accuracy and Time of the vFDS

In this section, we evaluate detection accuracy and response times of vFDS. Note that with the proper setting of threshold values, there will be no false alarm (that is, false positive) under normal conditions. We define the detection probability as the percentage of the successful identified attack instances over the total launched attacks.

7.1 Detecting SYN Flooding Attacks

We use a proprietary SYN flooder program to generate SYN flood attack traffic as done so in the previous experiment reported in [7]. The quoted experiment showed that a minimum of 500 SYNs per second is required to overwhelm a server.

Our detection mechanism is designed to work with a lower bound of flooding attacks of 500 SYNs per second. We have used the DEC trace as the normal background traffic (see Fig. 4). The flooding traffic of various rates (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 experiments, it has been empirically observed that the setting $X = 10$ and $\eta = 1$ is sufficient to capture all significant deviations in protocol attribute behaviors. The simulation results for different flooding rates are listed in Table 1.

Fig. 11 shows the estimated threshold HD (with $X = 10$ and $\eta = 1$) along with the observed HD for the DEC trace.

TABLE 1
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%

The measured distances for the DEC trace is always smaller than the estimated threshold. An alert flag is raised only when the observed HD of a particular testing period becomes higher than that of the estimated threshold HD in that period. Fig. 11 also shows how the observed HD dramatically changed 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 HD shoots up to 0.668, easily crossing the threshold value and subsequently raising an alert.

7.2 Detecting SIP Flooding Attacks

In this experiment, we defend a SIP proxy server against INVITE flooding attacks. The iSoftTech SIP Proxy Server [2] running on a Linux PC (Pentium 3, 1 GHz) and the CISCO SIP Proxy Server [1]—two popular commercial products—are expected to 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.

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. Our intent is to verify that with the appropriate setting of the threshold value, vFDS not only identifies the flooding attack of 500 INVITEs per second with an accuracy of 100 percent 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

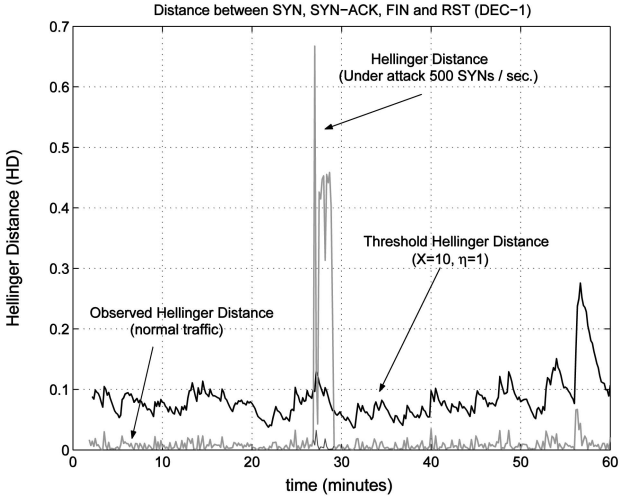


Fig. 11. Observed and threshold HDs (DEC).

TABLE 2
SIP INVITE Flooding Detection Performance of vFDS
$(n = 10, \triangle t = 10 \text{ seconds}, X = 20, \eta = 1)$

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%

distributed between 10 to 55 minutes. The experimental results for different flooding rates are listed in Table 2.

The HD plotted in Fig. 9 shows a maximum observed distance of $8 * 10^{-3}$ and an average distance of $0.9 * 10^{-3}$. Therefore, by setting the threshold distance with $X = 20$ and $\eta = 1$, any significant deviation in the SIP traffic is detected without raising a false alarm.

Fig. 12 illustrates the dynamics of the estimated threshold HD and the observed HD. Because the spikes of the observed HD are much smaller than those of the estimated threshold distance, no false alarm is raised. Injecting attack traffic consisting of 500 INVITEs per second starting at time 29.833 minutes causes the sudden surge of the observed HD during the next testing period, reaching 0.3597. Because the observed HD during the attack period is much higher than the average threshold distance of 0.02, an alert flag is raised.

7.3 Detecting RTP Flooding Attacks

Our attack traffic for RTP floods are generated as similar as the experiment performed by Qovia Inc. [24], where SIP-based *Siemens Optiplus 600* phones with a G.711 codec perform well at 500 RTP PPS with a packet size of 200 bytes, but as the RTP packet rate increases to 2,500 PPS, the voice quality significantly deteriorates, and subsequently, the connection breaks. Consequently, we assume that UACs use G.711 (that is, 50 PPS) codec algorithm with an RTP traffic rate of 500 PPS or more to create an RTP flood.

As described in Section 5.3, for incoming voice streams $i \leq N$, we use two media attributes $n^i_{theoretical}$ and $n^i_{observed}$ for each stream. Then, we compute the total number of expected incoming RTP packets to be $\sum_{i=1}^{N} n^i_{theoretical}$, and

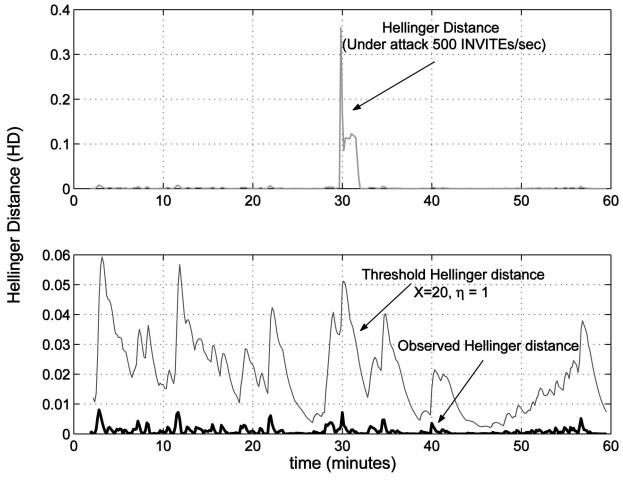


Fig. 12. Observed and threshold HDs (SIP).

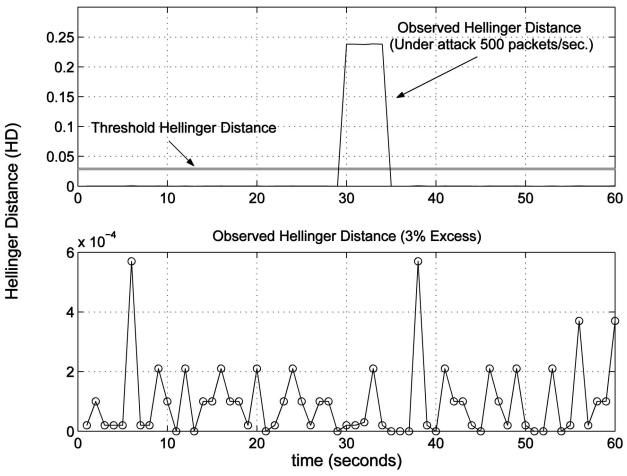


Fig. 13. Observed and threshold HDs (RTP).

consider any significant deviation from this expected value to be an RTP attack.

In order to detect RTP floods, we compute another attribute $n^i_{threshold}$, an upper bound for the tolerable number of RTP packets per second. Thus, the tolerable upper bound for the total number of RTP packets is $\sum_{i=1}^{N} n^i_{threshold}$. After that, we use the HD method to quantify the difference between these two sums of attributes experimentally and set the threshold as a *static value* based on the quantified result, as opposed to the dynamic threshold computation used earlier, because RTP streams are stable and only depend upon the codec and its rates.

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 HD is computed as

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

Fig. 13 plots the observed and threshold HDs for the voice stream. The flooding traffic of 500 PPS is injected into the voice stream 29 seconds from the start of the stream and lasts for 5 seconds. As seen in the figure, the measured distances (with $\triangle t = 1$ second) of the RTP stream under flooding attack are 10 times higher than the threshold distance, and hence, an alarm is raised.

7.4 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 percent of them have detection times between 13-18 seconds, and for the remaining 16 percent of the experiments, their detection times are 10 seconds. In both set of experiments, our testing period (that is, $\triangle t$) is fixed at 10 seconds. In the

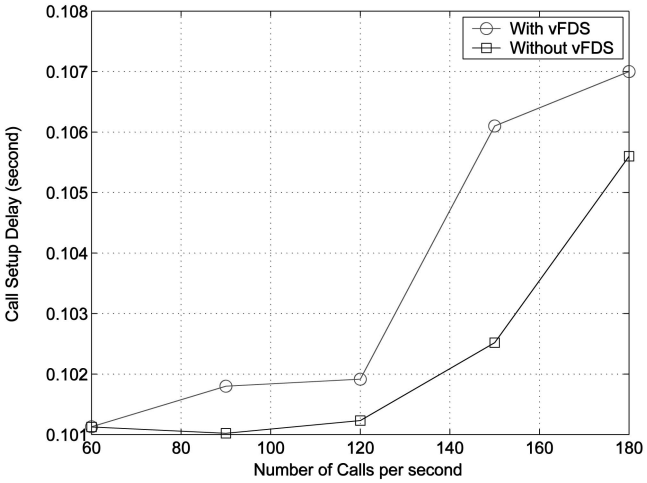


Fig. 14. Call setup delay.

RTP flooding attacks, the testing period is 1 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.

8 Impact of the vFDS on the Quality of a VoIP Service

In traditional telephony, performance requirements are generally expressed as cross-switching times or message transfer times, assuming that the voice quality is acceptable. However, because VoIP is an application running on the TCP/IP stack, we need to consider the effect of monitoring on the voice quality as well, of which jitter is the most prominent attribute.

8.1 Call Setup Delay

Because the primary use of IP telephony is to satisfy customers, any security mechanism that introduces long connection delays may not be adopted by the service providers. Consequently, the performance metric in which we are most interested is the extra delays induced to call setup times by the online placement of vFDS. The implementation of vFDS is based on Netfilter [22]. Netfilter provides a set of *hooks* in the Linux kernel's network protocol stack, allowing various modules to work with network packets. In general, call setup delay is defined as the interval between entering the last dialed digit and receiving ringback [10]. In SIP-based VoIP systems, the call setup time can be taken as the time interval between a caller sending an `INVITE` message and receiving a `180 ringing` message back from the callee. Fig. 14 shows the call setup delays with and without vFDS, respectively. It is evident that even with a high load of VoIP calls at the server, customers will not experience any noticeable call setup delays.

8.2 Effect on the Voice Quality

The detection of RTP flooding is based on the counts of RTP packets per second in a particular voice stream. The detection mechanism has a negligible effect upon RTP delay and jitter. Consequently, the voice quality perceived by the subscribers remains unaffected.

9 Related Work

The works by Wang et al. [35], [36] and Reynolds and Ghosal [25] 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 and Ghosal proposed the 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 the (`INVITE`, `200 OK`) pair to detect IP telephony call request flood attacks. Although the (`INVITE`, `200 OK`) pair is useful in detecting flooding attacks that originated inside the enterprise network, its usage for detecting flooding attacks that originated from the outside of the enterprise network is questionable. TLAS is based on the TCP behavior of (`SYN`, `ACK`) pairs. We do not use `ACK` packets in flooding detection, because it requires the state maintenance of a TCP session and more processing power to distinguish these ACKs for control packets from those ACKs for data packets. Wu et al. [38] proposed SCIDIVE, a stateful cross-protocol intrusion detection mechanism for VoIP.

There are many other commercial network security products, which take a similar approach to validating the observed traffic behavior against the expected traffic behavior. *Mazu Profiler* [20] 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.

Recently, Chen [6] developed a VoIP DoS attack detection system that maintains a state table for INVITE and non-INVITE transactions. The author claimed that state maintenance is a viable way to protect a SIP proxy server from DDoS attacks. Besides the DDoS attacks, many other VoIP security issues and threats have been discussed in [15] and [27].

HD is well studied in statistics and probability. It is a valuable metric in product measures and pointwise differentiability in some asymptotic problems [23]. It is often used in machine learning and many other applications, such as regression, measuring ecological distances [37], viral email propagation [32], and data swapping [13].

10 Conclusions and Future Work

`SYN`, `INVITE`, and RTP packet floods pose a serious threat to the IP telephony infrastructure. The multiprotocol-based VoIP service needs a fast and generic detection mechanism working across different protocol layers. We investigate the protocol attribute behaviors and characterize the network traffic with respect to the intrinsic correlation among protocol attributes. Utilizing HD, 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 Internet traces collected at exchange points of the Internet and the VoIP traces generated on an experimental SIP-based testbed.

Our experimental results show that vFDS can achieve high detection accuracy with a short detection time of 1-2 testing periods. In the future work, we plan to further improve the detection sensitivity of vFDS against low-rate attacks that span a longer period of time and conduct more exhaustive performance evaluations using diverse VoIP traces.

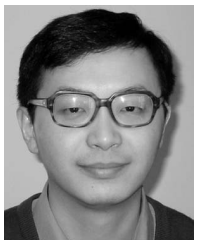
ACKNOWLEDGMENTS

An earlier version of this paper [29] was presented in the *Proceedings of the 14th International Workshop on Quality of Service (IWQoS '06)*. This work was supported in part by the US National Science Foundation under Grants CT-0627493 and CT-0627340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

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] "iSoftTech SIP Proxy Server," *Software Design Overview Template,* www.isofttech.com/downloads/SIP_3261_Proxy_Stack.pdf, 2005.
[3] AltiGen Communications, *AltiGen Alti-IP 600H IP Telephone,* Product Overview—VoIP Phones, http://www.altigen.com/analog-IP-telephone-sets.html, 2005.
[4] Arbor Networks, *Arbor Peakflow and Netflow,* Product Overview, http://www.arbornetworks.com/downloads/, 2006.
[5] M. Carson and D. Santay, *NIST Net Network Emulation Package,* Nist Net Web Site, http://snad.ncsl.nist.gov/itg/nistnet/, June 1998.
[6] E. Chen, "Detecting Dos Attacks on Sip Systems," *Proc. IEEE First Workshop VoIP Management and Security (VoIP MaSe '06),* Apr. 2006.
[7] T. Darmohray and R. Oliver, "Hot Spares for DoS Attacks," *;login: The Magazine of Usenix and SAGE,* vol. 25, no. 7, July 2000.
[8] DEC, "Digital Equipment Corporation Traces," *Hourly Traffic Traces,* 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, Univ. of Montreal, 2003.
[10] T. Eyers and H. Schulzrinne, "Predicting Internet Telephony Call Setup Delay," *Proc. First IP-Telephony Workshop (IPtel '00),* Apr. 2000.
[11] M. Fannes and P. Spincemaille, *The Mutual Affinity of Random Measures,* eprint arXiv:math-ph/0112034, Dec. 2001.
[12] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol—HTTP1.1,* IETF RFC 2616, 1999.
[13] S. Gomatam, A.F. Karr, C. Liu, and A.P. Sanil, "Data Swapping: A Risk-Utility Framework and Web Service Implementation," *Proc. Nat'l Conf. Digital Government Research (DG.O),* 2003.
[14] M. Handley and V. Jacobson, *SDP: Session Description Protocol,* IETF RFC 2327, 1998.
[15] A. Hoffmann, "Securing Large Scale VoIP Infrastructures," *Proc. Third Ann. VoIP Security Workshop,* June 2006.
[16] V. Jacobson and M.J. Karels, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM '88,* pp. 314-329, Aug. 1988.
[17] A.B. Johnston, *SIP Understanding the Session Initiation Protocol,* second ed. Artech House, 2004.
[18] G. Jongbloed and G. Koole, "Managing Uncertainty in Call Centers Using Poisson Mixtures," *Applied Stochastic Models in Business and Industry,* vol. 17, pp. 307-318, 2001.
[19] J. Klensin, *Simple Mail Transfer Protocol,* IETF RFC 2821, 2001.
[20] Mazu Networks, "Mazu Profiler," *Product Overview,* http://www.mazunetworks.com/resources/product-sheets/, 2006.
[21] NLANR, *NLANR Network Traffic Traces,* Front Range GigaPOP, Daily traffic traces, http://pma.nlanr.net/Traces/, 2005.
[22] P. Russell, *Netfilter/iptables,* Firewall, http://www.netfilter.org/, 2005.
[23] D. Pollard, *Asymptopia,* first ed., book in progress, http://www.stat.yale.edu/ pollard/, 2000.
[24] Qovia Inc., "Network Intrusion and QoS Impact in VoIP," white paper, http://www.qovia.com/, Aug. 2004.
[25] B. Reynolds and D. Ghosal, "Secure IP Telephony Using Multi-Layered Protection," *Proc. Network and Distributed System Security Symp. (NDSS '03),* Feb. 2003.
[26] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol,* IETF RFC 3261, 2002.
[27] H. Scholz, "Attacking VoIP Networks," *Proc. Third Ann. VoIP Security Workshop,* June 2006.
[28] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications,* IETF RFC 1889, 1996.
[29] H. Sengar, H. Wang, D. Wijesekera, and S. Jajodia, "Fast Detection of Denial of Service Attacks on IP Telephony," *Proc. 14th Int'l Workshop Quality of Service (IWQoS '06),* June 2006.
[30] H. Sengar, D. Wijesekera, H. Wang, and S. Jajodia, "VoIP Intrusion Detection through Interacting Protocol State Machines," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '06),* June 2006.
[31] W. Stevens, *TCP/IP Illustrated Volume-1,* first ed. Addison-Wesley, 1994.
[32] S.J. Stolfo, W.-J. Li, S. Hershkop, K. Wang, C.-W. Hu, and O. Nimeskern, "Detecting Viral Propagations Using Email Behavior Profiles," *ACM Trans. Internet Technology,* May 2004.
[33] Telecost, *Telecost: On Call Durations,* Product Overview, http://www.telecost.co.uk/Products/OnCallDurations.htm, 2005.
[34] K. Thompson, G.J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network,* vol. 11, Nov./Dec. 1997.
[35] H. Wang, D. Zhang, and K.G. Shin, "Detecting SYN Flooding Attacks," *Proc. IEEE INFOCOM '02,* June 2002.
[36] H. Wang, D. Zhang, and K.G. Shin, "SYN-Dog: Sniffing SYN Flooding Sources," *Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS '02),* July 2002.
[37] "World Agroforestry Center," *Regression and Analysis of Variance,* Tutorial, http://www.worldagroforestry.org/, 2005.
[38] 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," *Proc. Int'l Dependable Systems and Networks Conf. (DSN '04),* June 2004.
[39] Zultys Technologies, "Datasheet-ZIP 4X4," *Product Overview—VoIP Phones,* http://www.zultystechnologies.com, 2005.

Hemant Sengar received the BTech degree from Indian Institute of Technology, Kanpur, and the MS degree from George Mason University, Fairfax, Virginia. He is a PhD candidate in the Center for Secure Information Systems, Department of Information and Software Engineering, George Mason University. His current research interests are in the area of IP telephony and telecommunication networks security. He is a student member of the IEEE.



Haining Wang received the PhD degree in computer science and engineering from the University of Michigan, Ann Arbor, in 2003. He is an assistant professor of computer science at the College of William and Mary, Williamsburg, Virginia. His research interests lie in the area of networking, security, and distributed computing. He is particularly interested in network security and network quality of service (QoS) to support secure and service-differentiated internetworking. He is a member of the IEEE.

Duminda Wijesekera received the doctorate in Logic from Cornell University in 1990 and the doctorate in computer science from the University of Minnesota in 1997. He is an associate professor in the Department of Information and Software Engineering, George Mason University (GMU), Fairfax, Virginia. He holds courtesy appointments at the Center for Secure Information Systems (CSIS) and the Center for Command, Control and Coordination (C4I) at George Mason University, and the Potomac Institute of Policy Studies, Arlington, Virginia Prior to joining GMU, he was at Honeywell Military Avionics, Army High Performance Research Center, University of Minnesota, and the University of Wisconsin. During various times, his research interests have been in security, multimedia, networks, secure signaling (telecoms, railway, and SCADA), avionics, missile systems, Web, and theoretical computer science. He is a senior member of the IEEE.

Sushil Jajodia is university professor and BDM International professor of information technology and the director of the Center for Secure Information Systems at George Mason University, Fairfax, Virginia. He served as the chair of the Department of Information and Software Engineering from 1998 to 2002. His research interests include information security, temporal databases, and replicated databases. He has authored six books, edited 27 books and conference proceedings, and published more than 300 technical papers in the refereed journals and conference proceedings. He is the founding editor in chief of the *Journal of Computer Security* and is on the editorial boards of *IEE Proceedings on Information Security*, *International Journal of Cooperative Information Systems*, and *International Journal of Information and Computer Security*. He is a senior member of the IEEE and the IEEE Computer Society.