SIP-based VoIP Traffic Behavior Profiling and Its Applications

Hun Jeong Kang, Zhi-Li Zhang
University of Minnesota
Email: {hkang, zhzhang}@cs.umn.edu

Supranamaya Ranjan, and Antonio Nucci
NARUS
Email: {soups, anucci}@narus.com

Abstract

With the widespread adoption of SIP-based VoIP, understanding the characteristics of SIP traffic behavior is critical to problem diagnosis and security protection of VoIP services – two key aspects of providing dependable VoIP services. In this paper we propose a general methodology for profiling SIP-based VoIP traffic behavior at several levels: SIP server host, server entity (e.g., registrar and call proxy) and individual user levels – to derive "normal" behavior profiles. Using SIP traffic traces captured in a production VoIP network, we illustrate the characteristics of SIP-based VoIP traffic behavior in an operational environment and demonstrate the effectiveness of our general profiling methodology. Based upon the profiling methodology, we develop a simple and yet effective entropy-based anomaly detection algorithm for detecting potential security attacks as well as performance problems. We demonstrate the efficacy of our algorithm in detecting potential VoIP attacks through testbed experimentation.

1 Introduction

Voice over IP (VoIP) allows users to make phone calls over the Internet, or any other IP network, using the packet switched network as a transmission medium rather than the traditional circuit transmissions of the Public Switched Telephone Network (PSTN). VoIP has come a long way since it first rudimentary applications provided erratic yet free phone calls over the unmanaged Internet. VoIP technology has reached a point of being comparable in terms of grade voice quality with traditional PSTN yet consuming only a fraction of the bandwidth required by TDM networks. The maturity of VoIP standards and quality of service (QoS) on IP networks opens up new possibilities for carrier applications. Consolidation of voice and data on one network maximizes network efficiency, streamlines the network architecture, reduce capital and operational costs, and opens up new service opportunities. At the same time, VoIP enables new multimedia service opportunities, such as Web-enabled multimedia conferencing, unified messaging, etc, while being much cheaper.

VoIP offers compelling advantages but it also presents a security paradox. The very openness and ubiquity that make IP networks such powerful infrastructures also make them a liability. Risks include Denial of Service (DoS), Service Theft, Unauthorized Call Monitoring, Call Routing Manipulation, Identity Theft and Impersonation, among others. Not only does VoIP inherit all data security risks, but it introduces new vehicles for threats related to the plethora of new emerging VoIP protocols that have yet to undergo detailed security analysis and scrutiny.

But just how serious are the threats posed to VoIP [4]. Recently, there have been a string of attacks against either the VoIP infrastructure or end users. In one such incident, early June of 2006, two men were arrested for fraudulently routing approximately $500,000$ calls illegally over the VoIP network belonging to Net2Phone, a Newark, N.J., VoIP provider. Fifteen Internet phone companies were reportedly as the victims of this attack. More recently, ISS posted a report about a Denial-of-Service vulnerability in the IAX2 implementation of Asterisk, an open source software PBX. This vulnerability relates to the amount of time that a pending (but not yet authenticated) call is allowed to exist in memory on the server. New terms start to be coined over time just for VoIP attacks; "Vishing", is now used for phishing attacks using VoIP technology, or "Spit", now used for spam over VoIP.

Hence it is imperative for service providers to widely deploy scalable monitoring systems with powerful tools across their entire infrastructures such to robustly shield their VoIP infrastructure and protect their service, thereby providing *dependable* VoIP services. In this paper we propose a SIP traffic behavior profiling methodology, with the objective to identify anomalies and help service providers to accurately diagnose problems on-fly and promptly detect and trace-back on-going attacks on critical VoIP service (and their infrastructure). We propose a *multi-level*, *progressively refined* methodology that characterizes VoIP service activity in real-time by extracting and profiling a large variety of traffic features and metrics at three differ-

ent levels: (i)*server*, e.g. broad-view of their behavior by monitoring and keeping statistics related to only the message types (request vs response); (ii) *entity* , e.g. coarse-view of the servers activity by separating their logical roles into *registrar* and *call proxy*; and (iii) *individual users*, e.g. narrow-view of individual user activities, like typical average duration and length of the calls, number of calls received and made, etc. As a consequence, our methodology allows us to balance the speed of profiling, the resource consumption, the desired sophistication of behavior characteristics, and finally the level of security to be offered, based on the specific objectives and needs of the VoIP operator. Built upon the SIP traffic behavior profiling methodology, we develop a simple and yet effective entropy-based anomaly detection algorithm for detecting potential security attacks as well as performance problems. We demonstrate the efficacy of our algorithm in detecting potential VoIP attacks through testbed experimentation.

The remainder of this paper is structured as follows. In Section 2 we introduce some basic concepts of the SIP-based service, we discuss how challenging is to monitor and profile the service and introduce the data sets used in this paper to identify meaningfully VoIP traffic features to be profiled. In Section 3 we present in great details our methodology. First we introduce a new algorithm to automatically discover SIP servers, and breaks down their logical functionality, e.g. registrars and call proxies. Second we discuss at very high-level which traffic features should be monitored at the *server*, *entity* and *individual user* levels in order to gain a complete view of the VoIP traffic activity. Third we introduce a new algorithm based on Information Entropy that profiles the chosen metrics over time and generates alerts while any of the features diverges from its historical trend. Section 4 analyzes the traffic features discussed in Section 3 while using real packet traces collected from a wireless VoIP service provider. This section highlights some preliminary interesting findings that validates the overall approach. Section 5 describes three different VoIP attacks generated in a controlled lab environment and presents the outcome results of applying our methodology. Section 7 summarizes our findings and concludes the paper.

2 Background and Data Sets

We first provide a quick overview of SIP-based IP telephony. We then briefly touch on the challenges in profiling SIP traffic behaviors based on *passive packet monitoring*, and describe the SIP data sets used in our study.

2.1 SIP-based VoIP Service

The session initiation protocol (SIP) [9] is the Internet standard signaling protocol for setting up, controlling, and terminating VoIP sessions[1]. SIP-based VoIP services require *infrastructure* support from entities such as SIP registrars, call proxies, and so forth (see Fig. 1) – we collectively refer to these entities as *SIP servers*. A SIP registrar associates SIP users (e.g., names or identities called *SIP URIs*) with their current locations (e.g., IP addresses). A SIP call proxy assists users in establishing calls (called *dialogs* in the SIP jargon) by handling and forwarding signaling messages among users (and other SIP servers). In practice, a physical host (SIP server) may assume multiple logical roles, e.g., functioning both as registrars and call proxies.

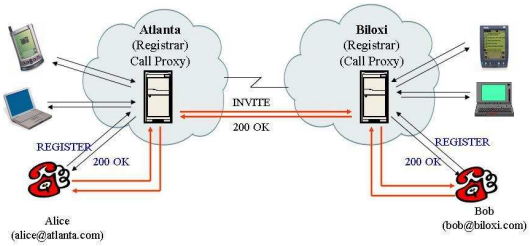


Figure 1. SIP servers and clients

SIP is a text-based `request-response` protocol, with syntax very similar to HTTP. SIP messages are of type either `request` or `response`. The `method` field is used to distinguish between different SIP operations. The most common `methods` include `REGISTER` (for user registration), `INVITE`, `ACK`, `BYE`, `CANCEL` (these four used for call set-up or tear-down), `SUBSCRIBE`, and `NOTIFY` (for event notification). `Response` messages contain a `response code` informing the results of the requested operations (e.g., `200 OK`). The `FROM` and `TO` fields in an SIP message contain respectively the SIP URIs of the user where a `request` message is originated from (e.g., the caller of a call) or destined to (e.g., the callee of a call). In the case of a `REGISTER` message, both `FROM` and `TO` typically contain the SIP URI of the user where the `request` is originated. Other important fields include `VIA` and various identifiers and tags to string together various transactions and dialogs. The reader is referred to [9] for details.

2.2 Problem Discussion and Data Sets

In this paper, we focus on characterizing and profiling SIP-based VoIP traffic behavior by using *passive traffic monitoring*, with the objective to identify anomalies to

[1] In addition to IP telephony, it can also be used for teleconferencing, presence, event notification, instant messaging, and other multimedia applications.

help diagnose problems and detect potential attacks on critical VoIP services (and their infrastructure). We assume that passive packet monitoring and capturing devices are deployed in the underlying network hosting VoIP services. In addition to the standard layer-3 (IP) and layer-4 (TCP/UDP) header information, portion of layer-7 payload containing appropriate application protocol (SIP) fields are also captured. The captured packet header and payload information is then processed and parsed for our analysis and profiling. Unlike the layer 3/4 header fields which generally have well-defined and *limited* semantics, the layer-7 application protocol such as SIP has a variety of fields, with *rich* semantics that are often context-sensitive and sometimes even implementation-specific. For example, with the SIP protocol itself, the meaning of the same fields may depend on the method used. Hence a major challenge in performing layer-7 protocol analysis and behavior profiling is to determine how to judiciously incorporate application-specific semantics or "domain knowledge" to select appropriate set of key features to capture the essential behavior characteristics of the application in question. In the next section we present such a general methodology for characterizing and profiling SIP-based VoIP traffic behavior.

Our profiling methodology is motivated and substantiated by in-depth analysis of SIP traffic traces captured in an operational network of a commercial wireless VoIP service provider. The results reported in this paper use three SIP traces from this network, referred to as *Trace I (13:55-14:30)*, *Trace II (19:00-19:40)* and *Trace III (19:55-20:30)*, respectively (the numbers within the parentheses indicate the start and end time of the traces). They are of about 40 minutes or so long, captured between 13:00 h and 21:00 h within a single day.

3 General Methodology

In this section, we present a multi-level profiling methodology for characterizing SIP traffic behavior using layer-3 to layer-7 protocol information obtained from *passive network monitoring*. In order to characterize and profile SIP server behaviors by using passively collected SIP traffic traces, we need to identify the IP addresses associated with SIP servers. We first introduce a simple heuristic for identifying the IP addresses associated with SIP servers (both SIP registrars and call proxies) based on passive monitoring of SIP traffic. We then present a general methodology for characterizing and profiling SIP server behaviors at multiple levels.

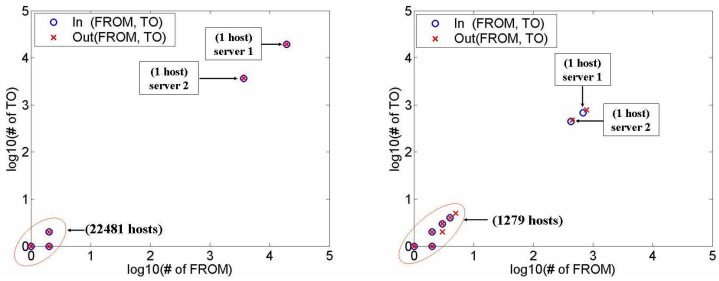
3.1 Discovering SIP Servers

The key observation behind our heuristics is based on the role of SIP servers in SIP-based VoIP communications: typically users must register with SIP registrars; and users' call signaling must get through SIP call proxies (see Fig. 1). Hence the IP address associated an SIP server will consistently see a large number of SIP messages going through it (i.e., with the said IP address as either the source or destination IP addresses); furthermore, we will also see a large number of distinct `FROM` and `TO` fields in the appropriate SIP messages (e.g., `INVITE`, `REGISTER`) associated with this IP address. The baseline algorithm for SIP call proxy discovery is given in Algorithm 1 examining the SIP `INVITE` messages. By examining the SIP `REGISTER` messages, we have a similar algorithm for SIP registrar discovery.

Algorithm 1 Baseline Algorithm for SIP Call Proxy Discovery

Parameters: message set M, threshold α;
Initialization: $IPSet := \emptyset$; $ProxyIP := \emptyset$;
for each $m \in M$ **do**
  if m.method == INVITE **then**
    x = m.sourceIP; y = m.destinationIP;
    $from$ = m.FROM; to = m.TO;
    if $x \notin IPSet$ **then**
      $x.Out_{\texttt{FROM}} = \{from\}$; $x.Out_{\texttt{TO}} = \{to\}$;
      $x.In_{\texttt{FROM}} = \emptyset$; $x.In_{\texttt{TO}} = \emptyset$;
    else
      $x.Out_{\texttt{FROM}} = x.Out_{\texttt{FROM}} \cup \{from\}$;
      $x.Out_{\texttt{TO}} = x.Out_{\texttt{TO}} \cup \{to\}$;
    end if
    if $[|x.In_{\texttt{FROM}}|, |x.In_{\texttt{TO}}|, |x.Out_{\texttt{FROM}}|, |x.Out_{\texttt{TO}}|] > [\alpha, \alpha, \alpha, \alpha]$ **then**
      $ProxyIP = ProxyIP \cup \{x\}$
    end if
    if $y \notin IPSet$ **then**
      $y.In_{\texttt{FROM}} = \{from\}$; $x.In_{\texttt{TO}} = \{to\}$;
      $y.Out_{\texttt{FROM}} = \emptyset$; $y.Out_{\texttt{TO}} = \emptyset$;
    else
      $y.In_{\texttt{FROM}} = y.Out_{\texttt{FROM}} \cup \{from\}$;
      $y.In_{\texttt{TO}} = y.In_{\texttt{TO}} \cup \{to\}$;
    end if
    if $[|y.In_{\texttt{FROM}}|, |y.In_{\texttt{TO}}|, |y.Out_{\texttt{FROM}}|, |y.Out_{\texttt{TO}}|] > [\alpha, \alpha, \alpha, \alpha]$ **then**
      $ProxyIP = ProxyIP \cup \{y\}$
    end if
  end if
end for

In Algorithm 1, for each IP address a in the SIP messages (either as the source or destination IP) we maintain four records, $a.In_{\texttt{FROM}}$, $a.In_{\texttt{TO}}$, $a.Out_{\texttt{FROM}}$ and $a.Out_{\texttt{TO}}$, which maintain, respectively, the set of unique users (or rather their URIs) seen in the `FROM` and `TO` fields of the SIP `INVITE` messages *received* (In) by or *sent* (Out) from a. If the number of distinct users in each of the four records



(a) REGISTER (b) INVITE

Figure 2. Hosts corresponding (FROM,TO)

exceeds a threshold α [2] for an example, then a is included in the SIP call proxy candidate set $ProxyIP$. By ensuring the diversity of callers (FROM) and callees (TO) in both the SIP INVITE messages originating from and destined to a given IP, we minimize the chance of misclassifying of a user in the *forward* mode in which incoming INVITE messages are forwarded to another location, or similarly, when a user is in a conference mode. In both cases, the TO field of the INVITE messages will contains the URI (or its variants) of the forwarder. Hence the size of corresponding $In_{\texttt{TO}}$ and $Out_{\texttt{TO}}$ will be small. We have extended the baseline algorithm to incorporate additional mechanism to address the effect of NAT boxes and other issues, the details of which can be found in [5].

In the following we illustrate the effectiveness of our baseline algorithm using the real SIP traffic traces. Fig. 2(a) shows the number of unique FROM's vs. TO's in the SIP REGISTER messages *received* (i.e., $In_{\texttt{FROM}}$ vs. $In_{\texttt{TO}}$) and *sent* (i.e., $Out_{\texttt{FROM}}$ vs. $Out_{\texttt{TO}}$) by each IP address seen in the SIP traces. Similarly, Fig. 2(b) shows the number of unique FROM's vs. TO's in the SIP INVITE messages *received* and *sent* by each IP address seen in Trace II. Note that as many hosts (i.e., IP addresses) may have the same number of FROM's and TO's (the labels on the side indicate the number of such hosts). In both cases, only two IP addresses (which are the same two IP entity addresses in Fig. 2(a) and (b)) have significantly more FROM's and TO's than the remaining IP addresses, which have only one or very few distinct FROM's and TO's in a 30-40 minute time interval. These two IP addresses are those of two SIP servers (in this case functioning both are registrars and call proxies) in the network, one serving more users than the other in this 40 segment SIP trace[3]. Hence our baseline algorithm can effectively identify the IP addresses associated with the SIP severs (registrars or call proxies) by appropriately setting the threshold (e.g., $\alpha = 100$).

[2]The threshold can be determined, for example, by first plotting $In_{\texttt{FROM}}$ vs. $In_{\texttt{TO}}$ and $Out_{\texttt{FROM}}$ vs. $Out_{\texttt{TO}}$ in a scatter plot, see Fig. 2.

[3]In the Trace III we see that the role of the two SIP servers is reversed, with the latter server serving more users than the former one.

3.2 Profiling SIP Server and User Behaviors

Once we have identified the IP addresses associated with the SIP servers, we characterize and profile the behavior of SIP servers by examining the SIP messages going through them. We characterize and profile the behaviors of SIP servers (and their associated users) at three levels – *server host*, *server entity* and *(individual) user* – by introducing a range of features and metrics from coarser granularity and finer granularity in terms of the amount of application-specific (i.e., SIP) semantic information. This *multi-level, progressively refined* methodology allows us to balance the speed of profiling, resources required, desired sophistication of behavior characteristics, and level of security, an so forth based on the objectives and needs of a SIP-based VoIP operator.

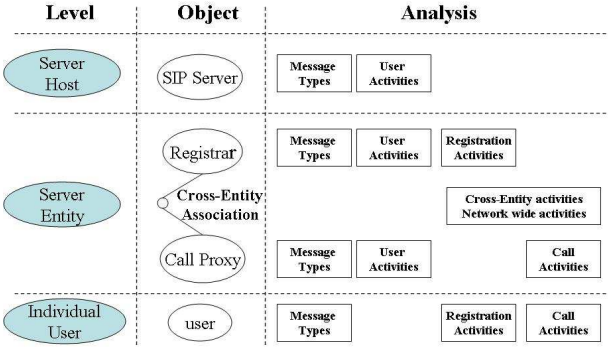


Figure 3. Multilevel Profiling

Fig. 3 is a schematic depiction of our multi-level profiling methodology. At the *server host* level we maintain only aggregate features and metrics to provide a broad view of a SIP server behavior and its "health" by examining only the message types (request vs. response) into and out of a SIP server and extracting only coarse-grain user statistics information. At the *server entity* level, we separate the (logical) role of a SIP server into *registrar* and *call proxy*, as these two separate entities require a different set of features and metrics to characterize their respective behavior. Based on the SIP semantics, we examine the *method* field of a SIP message to attribute it to either the SIP registrar or call proxy (e.g., a SIP REGISTER messages and its response are part of a registrar activity while a SIP INVITE message and its response are part of a call proxy activity), and compute appropriate features and metrics for the corresponding registrars and call proxies. We also cross-examine the activities of SIP registrars and call proxies to build cross-entity associations. At the (individual) user level, we attribute the SIP messages to individual users, and maintain statistics and features to characterize individual user behaviors. In the following we provide a more detailed description of

our multi-level profiling methodology.

a. Server Host Level Characterization.

We characterize the aggregate behavior of a SIP server by maintaining two types of (aggregate) statistics and features: i) we count the number of `request` and `response` messages received (i.e., *fan-in*) and sent (i.e., *fan-out*) by each SIP server (and derivatively their corresponding ratios) over a given period of time T (say, 5 or 15 minutes); ii) we count the number of unique users (URIs) seen in the `FROM` and `TO` fields of SIP `request` messages, and compute an aggregate *user activity diversity* (*UAD in short*) metric from the distribution of such data over T. This UAD metric is computed as follows: Let m be the total number of SIP `request` messages over T, and n is the total number of distinct users seen in the message. For each unique user i, m_i is the number of SIP messages with i in either the `FROM` or `TO` field of the messages. Then $p_i = m_i/m$ is the frequency that user i is seen in the SIP messages. The user activity diversity metric, UAD, is then given by

$$UAD := (-\sum_i p_i \log p_i)/\log m \in [0, 1], \quad (1)$$

where the numerator is the entropy of the distribution $\{p_i\}$ while the $\log m$ is its maximum entropy – the ratio of the two is the standardized entropy (or *relative uncertainty*). UAD thus provides a measure of "randomness" of user activities as captured by the distribution $\{p_i\}$: for $n >> 1$, if $p_i \approx 0$, a few users dominate the SIP activities (in other words, they appear in most of the messages), whereas $p_i \approx 1$ implies that $p_i = O(1/m)$ and thus each user only appears in a few number of SIP messages (hence overall the user activities appear random).

b. Server Entity Level Characterization.

Registrar: Using the `method` field of SIP messages, we separate registrar-related messages (e.g., the `REGISTER` messages and their responses) and use them to generate statistics and features for registrar behavior profiling. Similar to the server level analysis, we maintain *aggregate statistics* regarding the number (and ratios) of `REGISTER` and other registrar-related requests and responses received and sent by a registrar. In terms of *user activities*, we maintain the number (and list) of users that are successfully registered, and compute a similar user activity diversity (UAD) metric with respect to the registrar. In addition to these aggregate statistics and features regarding the message types and user activities, we also perform more detailed registration analysis. We examine the response codes in the response messages to maintain statistics about the number of *successful* and *failed* registrations. We also calculate the registration periods of users (i.e., the time lapses between two consecutive `REGISTER` messages from the same user) and the inter-arrival times of any two consecutive `REGISTER request` messages (from different users). From the former we compute the (average) registration period of the registrar and from the latter we derive a (fitted) model for the user `REGISTER request` arrival process. Together, they not only reveal the configuration of the registrar but also the temporal behavior of the registrar.

Call Proxy: By analyzing the SIP messages related to call activities (e.g., SIP messages with the `INVITE`, `BYE` methods and their responses), we generate statistics and features for call proxy behavior profiling. Similar as before, we maintain aggregate statistics regarding the numbers and ratios of various call requests (`INVITE`, `BYE`, `CANCEL`, etc.) and their responses received and sent by a registrar. We maintain several user activity diversity (UAD) metrics regarding the aggregate user call activities: UAD_{caller}, UAD_{callee} and $UAD_{\text{caller-callee}}$, which measure the UAD of callers, callees and caller-callee pairs. Each of these metrics is computed using equation (1) with appropriate defined parameters: m is the number of SIP call request messages (SIP `INVITE`, `BYE` and `CANCEL`) requests, and i) for UAD_{caller}, m_i is the number of SIP call request messages with user i in the `FROM` field, ii) for UAD_{callee}, m_i is the number of SIP call request messages with user i in the `TO` field, and iii) for $UAD_{\text{caller-callee}}$, we replace m_i by m_{ij} where m_{ij} is the number of SIP call request messages with user i in the `FROM` field and user j in the `TO` field.

Furthermore, we perform a more detailed call analysis to maintain various call statistics and features of a call proxy. These include the number of on-going calls, completed calls (calls ended by `BYE` only), canceled calls (calls ended by `CANCEL` only), *failed* calls (calls receiving a response with a `Request Failure (400-499)` response code), and so forth, in a given time period. We also compute statistics (average, standard deviation or distribution) regarding call durations and call request arrival rates.

Cross-Entity Association: we also correlate statistics and features to generate a cross-entity and network-wide view of the SIP traffic. The detailed description is provided in [5] due to space limitation.

c. Individual User Level Characterization.

If needed, we can also maintain statistics and features regarding the individual user activities. For example, from the user call activities we can maintain the (typical or average) number of calls made or receiver by each user u, and compute the diversity of callees ($UAD^{(u)}_{\text{callee}}$) of the calls made by the user as well as the diversity of callers ($UAD^{(u)}_{\text{caller}}$) of the calls received by the user u. Other statistics such as (average) call durations may also be maintained. Due to space limitation, we do not elaborate them here.

4 Characteristics of SIP Traffic Behavior

In this section, we apply the general profiling methodology presented in the previous section to analyze the SIP traces to illustrate the characteristics of SIP traffic in a real

VoIP network and use them to justify the statistics and features we have taken for profiling SIP traffic behavior. In particular, we show that in normal operational environments SIP traffic behavior tends to be very stable both in terms of various SIP message types, user registration, call, and other related activities.

4.1 Overall Server Level Characteristics

Throughout this section, we primarily use *TRACE II* and server-1 as an example to illustrate the results. Fig. 4(a) shows the numbers of `request` and `response` messages received (*REQin*, *RESin*) and sent (*REQout*, *RESout*) by server-1 over 5-minute time intervals of the SIP traces. We remove the first and last 5 minutes of the segment to avoid the boundary effect. Fig. 4(b) shows, respectively, the ratios of *REQin* vs. *RESout*, *REQout* vs. *RESin* and *REQin* vs. *REQout* over the same 5-minute time intervals. We see that overall the total numbers of `request` and `response` messages received and sent by the SIP server do not vary significantly. In particular, for every one `request` message received/sent by the SIP server, on the average there is approximately one `response` message sent/received by it – this is generally expected. There are roughly twice as many `request` messages received by the SIP server than sent by it. As we will see shortly, this is primarily due to the `REGISTER` messages which comprise a large portion of the total `request` messages received by the SIP server. Unlike many SIP `request` messages of other `methods` (e.g., `INVITE`), a `REGISTER request` message does not trigger the SIP server to generate another `request` message except a `response` message.

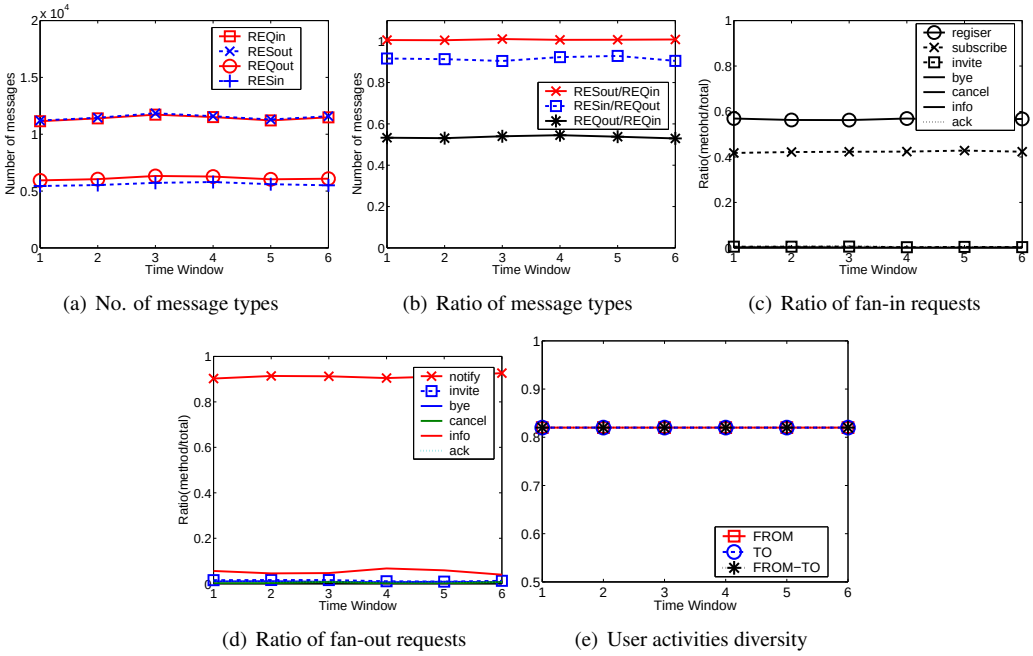
We break down the SIP `request` messages based on the `method` type and count their numbers over 5-minute time intervals. Figs. 4(c) and (d) shows the proportions of `request` messages of each `method` type received and sent respectively by the SIP server. As noted earlier, `REGISTER request` messages consist of nearly 60% of the total `request` messages received by the SIP server, while `SUBSCRIBE request` messages form 40% of them. In particular, there is no `NOTIFY request` messages received by the SIP server. In contrast, the `NOTIFY` messages comprise 90% of the total `request` messages sent by the SIP server, while there is no `REGISTER request` messages at all. More in-depth examination of the `SUBSCRIBE` messages received and `NOTIFY` messages sent by the SIP server reveals that there is approximately a one-to-one correspondence between the `SUBSCRIBE` messages received and `NOTIFY` sent: this is to be expected, as a `SUBSCRIBE` received by the SIP server would trigger one (and perhaps a few more) `NOTIFY` messages sent by the SIP server. In both the `request` messages received and sent by the SIP server, call-related SIP `request` messages such as `INVITE`, `BYE` and `CANCEL` consist of only a small portion of the total `request` messages received/sent by the server.

Fig. 4 (e) shows the user activity diversity (UAD) metric of the total SIP messages (both received and sent) by the SIP server over 5-minute time intervals, as well as those for SIP `request` messages received and sent separately. We see that the UAD metrics are close to 1 over all 5-minute time intervals and they are fairly stable. As seen in the next subsection this is primarily due to the periodic exchanges of the `REGISTER`, `SUBSCRIBE` and `NOTIFY request` messages and their `responses` between the SIP server and users. Our results show that the aggregate SIP traffic behavior is in general fairly stable and the aggregate statistics/features chosen in our profiling methodology provides a good summary of these stable characteristics. The same observations also hold true for server-2 (which handle a relatively smaller portion of SIP messages in *TRACE II*) as well as for *TRACE III* (where server-2 handles a large portion of SIP messages while server-1 handles a relatively smaller portion of them). *TRACE I*, on the other hand, contains an interesting *anomaly* which is detected by our profiling methodology. We will discuss and dissect this anomaly in more detail in Section 5.1.

4.2 Registrar Behavior Characteristics

We now focus on the `REGISTER request` messages and their responses of server-1 (functioning in the role of a registrar), and in particular, examining how `REGISTER` messages are generated by users. In Fig. 4(c) we have shown that `REGISTER` messages consist of 60% of the total `request` messages received by the SIP server (registrar). Moreover, the ratio of the number of `REGISTER request` messages vs. their responses is approximately 1. We observe that the user activity diversity metric for the `REGISTER request` messages is close to 1, indicating that there are no individual users who dominate the generation of `REGISTER` messages. The number of *unique* users seen in (the `FROM` field of) the `REGISTER` messages over given time intervals are examined. The total number of (distinct) users seen in *TRACE II* is 17800 that is almost the same to the number of users seen in 15-minute intervals and the number of users seen in a time interval of length T ($\leq$ 15 min) is roughly $17800 \times \frac{T}{15}$. As we will see, this is primarily due to registration periods and a `REGISTER` arrival process.

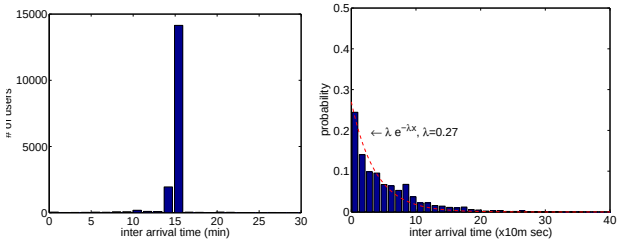
To further illustrate how `REGISTER` messages are generated, we calculate the time lapses between two consecutive `REGISTER` messages from each user, the distribution of which is shown in Fig. 5(a). The distribution clearly reveals that users generate `REGISTER` messages roughly periodically with a mean of 15 minutes. In Fig. 5(b) we plot



(a) No. of message types (b) Ratio of message types (c) Ratio of fan-in requests

(d) Ratio of fan-out requests (e) User activities diversity

Figure 4. Analysis on server behaviors



(a) Period of registration updates (b) Requests inter-arrival times

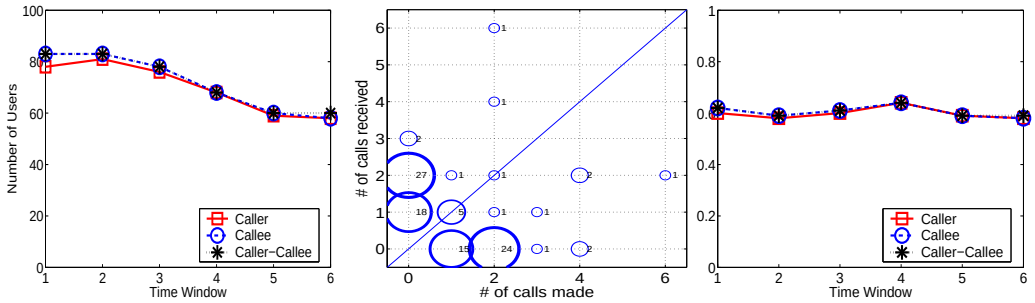
Figure 5. Analysis on registrar behaviors

the distribution of the *inter-arrival* times between two consecutive REGISTER messages (from two different users). The distribution can be well fit into an exponential distribution of the form $p(x) = \lambda e^{-\lambda x}$, where $\lambda = 0.27$. Hence we see that the number of REGISTER messages seen by the SIP server (registrar) follows approximately a Poisson process. We also study characteristics of SUBSCRIBE and NOTIFY messages which often follow the REGISTER messages of the same users. We perform the in-depth analysis on the number of message types, inter-arrival times between requests, the SUBSCRIBE periods from each user, and such results are included in [5]. The large number of messages and their regularity lead us to suspect that these SUBSCRIBE and NOTIFY messages are sent to subscribe and notify system resources such as voice mailboxes.

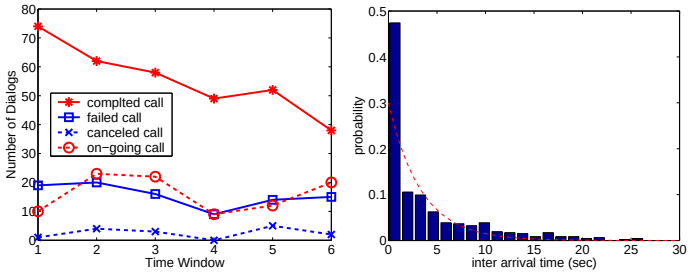
4.3 Call Proxy/User Call Behavior Characteristics

We now analyze characteristics of calls and call-related user activities. Comparing with the number of REGISTER, SUBSCRIBE and NOTIFY messages, we observe that call-related messages consist of a much smaller portion, indicating that while there are a large number of users (or more aptly, SIP phone devices) in the network, only a very small number of the users actually make phone calls in *a specific* period. This observation is further confirmed in Fig. 6(a) which shows the number of unique callers (users seen in the FROM field of INVITE messages) and callees (users seen in the TO field) *in each 5-minute intervals*. Recall that there are a total of 17800 unique users in the trace segment. Fig. 6(b) is a scatter plot showing the number of calls made vs. calls received per user over 5-minute intervals. Again we see that at individual user level, the numbers of calls made and received are generally very small and consistent. In terms of diversity of calls made by users, Fig. 6(c) shows the UAD metrics of callers (FROM's), callees (TO's) and caller-callee pairs (FROM−TO's) as defined in Section 3.2. We see that the call activities are fairly random, not dominated by any particular user (either as caller or callee).

The number of various call types (on-going, completed, failed, and canceled calls) over 5-minute intervals is shown in Fig. 6(d). We see that the number of calls in a typical 5-minute interval is fairly small, and the number of *failed* calls is relatively high due to user mobility or re-



(a) No. of callers and callees (b) Calls made vs received (c) User activities diversity



(d) Call types (e) Call inter-arrival times

Figure 6. Analysis on call proxy activities

ceiver statuses (busy or not available). Fig. 6(e) shows the call inter-arrival times are approximately exponentially distributed (the call arrival process is approximately Poisson). We observe that call duration typically lasts between 0-3 minutes while failed and canceled calls tend to last very short. Not surprising, these statistics are similar to traditional telephony, indicating that these call activities are human-generated.

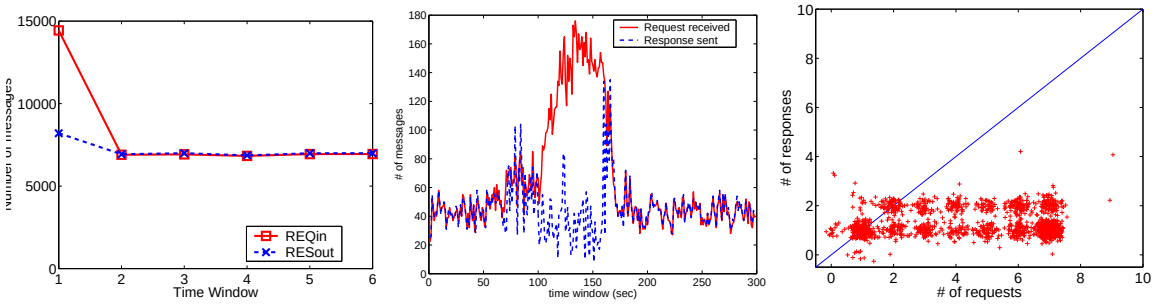
5 Applications

In this section, we illustrate the usefulness and applicability of our general profiling methodology in helping diagnose problems and detecting potential attacks against VoIP service and infrastructure through a case study as well as testbed experimentation. In particular, we develop a novel profiling-based feature anomaly detection algorithm for these purposes.

5.1 Problem Diagnosis: A Case Study

We use a case study – an interesting case uncovered in the analysis of the real-network SIP traces – to illustrate how our methodology can be used to detect and diagnose performance anomalies. As reported earlier, we see that overall the numbers of SIP `REGISTER` `request` and `response` messages and their ratios (over 5-minute intervals) stay fairly stable, and this can be mainly attributed to the fact that users generate `REGISTER` messages periodically and these messages are generated randomly from the users. These observations hold almost all 5-minute intervals for both servers in the traces except for one 5-minute interval of server-1 in *TRACE I*, where we have found an interesting "*anomaly*". As evident in Fig. 7(a), the number of `REGISTER` messages received by server-1 in the very first 5-minute interval in this trace segment is significantly larger than in other time intervals, and while the number of the responses sent by the server also increases slightly – in particular, the ratio of the numbers of `requests` vs. `responses` increases drastically.

To figure out the root-cause of this anomaly, we perform a in-depth analysis of the SIP messages in this anomalous 5-minute interval. Fig. 7(b) shows the number of `REGISTER` messages received by server-1 vs. the responses generated by it in each second of the anomalous 5-minute interval. We see that between around the 100th second to 160th second of this 5-minute interval, the number of `REGISTER` `requests` from users surges very quickly, while the `responses` returned by the server first dips for about 50-60 seconds before it surges as well, catching up with the number of `REGISTER` `requests`, after which everything returns to the norm. Fig. 7(c) is a scatter showing the number of `REGISTER` `requests` generated vs. number of `responses` received per user in the 1-minute time period from the 100th second to 160th second. To better illustrate the number of data points occupying a particular integer-valued grid (x, y), the data points are "perturbed"



(a) Message types (b) Total requests and responses (c) No. of requests and responses per user

Figure 7. Analysis on anomaly

slightly around it at random. We see that instead of the normal one `REGISTER request` and one `response` per user, many users send from 2-7 `REGISTER` requests while receiving one or two `responses`. Closer investigation reveals that the problem is caused by the SIP server not responding to the user registration requests immediately, triggering users to repeatedly re-transmit their requests within a few seconds until they either gives up or receive a response with either response code `404 Not Found`, `408 Request Timeout`, or (eventually) `200 OK`.

Since all these users were eventually able to successfully register with the SIP server, the surge of the `REGISTER requests` is unlikely caused by denial-of-service attacks with spoofed or frivolous `REGISTER` messages (as were originally suspected by us). That the SIP server failed to respond to the user registration requests in a timely fashion may be caused by delay or slow response from some remote (user/call) database with which the SIP server was interacting[4] This performance anomaly can be easily detected using a simple anomaly detection algorithm such as the one described below.

5.2 Feature Anomaly Detection and VoIP Attacks

Our profiling methodology produces an ensemble of statistics and features over time: for each statistics/feature, a time series is generated. Sudden changes or deviations from expected behavior in a subset of the statistics/feature time series signify anomalies. As illustrated later in this section, different VoIP attacks may trigger a different (sub)set of statistics/features to exhibit sudden changes or deviant behaviors. Our profiling-based anomaly detection algorithm consists of the following three key components/phases per-feature:

• **Baselining/Profiling:** In this phase, which lasts from $[0\text{-}T_{learn}]$ time windows, we baseline the feature and obtain an estimate of the maximum deviation possible under normal circumstances. We use two sliding windows, one for averaging the feature values and the other for averaging the instantaneous rate-of-change of the feature values, with the windows denoted as T_1 and T_2, where $T_1 + T_2 < T_{learn}$. The output of this phase is the maximum deviation of the rate-of-change from its average. Thus, more precisely: (1) First, for the feature averaging, we use an Exponential Weighted Moving Average (EWMA) with a $beta = \frac{2}{T_1+1}$ to obtain the feature average at time t as: $EWMA(t) = \beta EWMA(t-1) + (1-\beta)f(t)$; (2) Thus, we measure the instantaneous rate-of-change as: $s(t) = |\frac{f(t)}{EWMA(t-1))} - 1|$; (3) Over the window T_2, we average the instantaneous rates-of-change such that for every time t in the period $[(T_1 + T_2)\text{-}T_{learn}]$, we obtain the average slope $s_{avg}(t) = \frac{1}{T_2}\sum_{t-T_2}^{t} s(t)$; (4) At a time $t+1$, we calculate instantaneous deviation of rate-of-change from its average as: $d(t+1) = \frac{s(t+1)}{s_{avg}(t)}$. We learn the maximum of this deviation in the time period $[(T_1 + T_2 + 1)\text{-}T_{learn}]$ as: $\alpha = \max_{t=T_1+T_2+1}^{T_{learn}} d(t)$.

• **Alerting along with Lock-in/Lock-out of Averages:** After the learning period is over, we monitor the instantaneous slope $s(t)$ and if it is α times greater than the average slope $s_{avg}(t)$, then we raise an alert and increase the alert-level. Otherwise, we update the feature average $EWMA(t)$ and average slope $s_{avg}(t)$. Note that, if an alert was raised, then we *lock-in* the values for feature average and average slope, and these values are *locked-out* i.e., can be updated only when the alert value is reduced to zero. False alarms are avoided by having multiple levels of alerting and the system is said to be in an anomalous state only when the alert level is at its maximum. In particular, we use four levels: Green (alert=0), Yellow (alert=1), Orange (alert=2) and Red (alert=3).

• **Reporting:** This phase is used for reporting the sus-

[4]This problem points to a potential implementation flaw in the SIP client software: when a registration request times out, the client immediately retransmits the request, thereby causing a surge of requests and thus aggravating the problem. A better solution would have been to use an exponential back-off mechanism to handle the retransmission of the registration requests.

picious elements of a feature that are contributing to the anomaly. This is done only on the features that report the User Activity Diversity over the feature's distribution e.g., UAD_{caller}, UAD_{callee}. It is applied only when the alert level is updated to its maximum level.

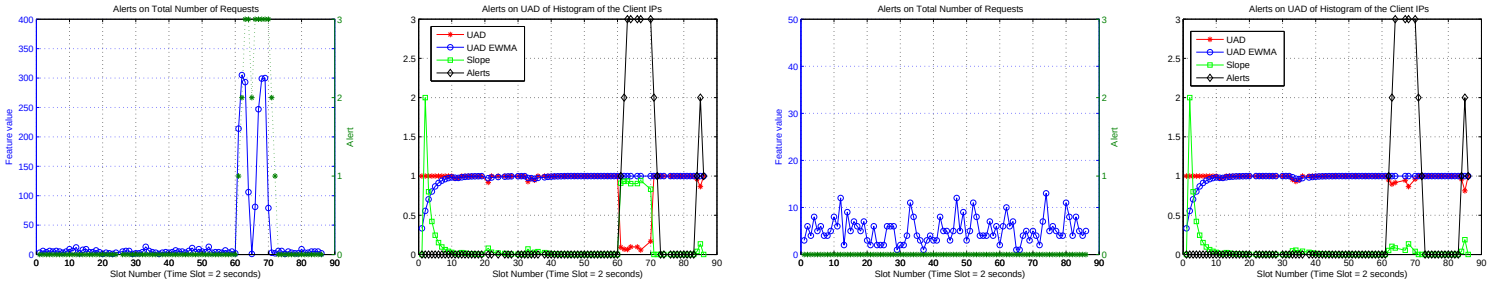
Before we proceed to report the performance of the above anomaly detection algorithm, we briefly discuss some common potential attacks on VoIP services, and what statistics/features in our SIP behavior profiles may provide alerts about such attacks. Given the generality of our multi-level profiling methodology, we can detect a wide variety of attacks: DoS and DDoS attacks on VoIP infrastructure or users, VoIP spam, and worms that exploit VoIP protocols to spread. The underlying hypothesis towards detecting these attacks is that such an attack introduces either a volume surge, a sudden change or deviation in the ratio/distribution statistics or metrics (e.g., randomness) in one or multiple features. Moreover, the ability to cross-correlate across multiple features also provides us the unique capability to classify the attack. Consider an example of call spam attacks, defined as when a spammer generates many calls, most likely in an automated fashion towards several unsuspecting callees (e.g., automated calls made by telemarketers simultaneously to many callees advertising a product). Thus, by varying the following parameters, a spammer can generate a variety of attacks: (1) number of callers per spammer IP address; (2) whether the IP addresses are legitimate or not; (3) number of IP addresses and; (4) volume of spam calls. The values that each of these features take can be subjectively categorized as "one (or few)" or "many". Thus, there are 2^4 unique spam attacks, two of which are (i) *high volume spam* where one caller per legitimate IP address sends large number of calls to random callees; (ii) and *low volume spam* where one caller per legitimate IP address sends moderate number of calls to random callees.

5.3 Experimental Testbed and Results

To validate the efficacy of our detecting algorithm, we use a testbed consisting of two machines, connected via an OC-12 link (1.5 Gbps). Each machine has two Intel(R) Xeon(TM) CPU 3.40GHz processors and 4 Gbytes memory available to a process running on the Linux kernel 2.4.21. We replay the trace by using *tcpreplay* from one machine while the other one is configured to sniff packets off-the-wire and runs our packet analyzer as well as the anomaly detection module. The packet analyzer parses layer-7 payloads off-the-wire from network links and emits an annotated vector per packet. Throughout the performance experiments, the capability of our packet analyzer is pegged at a line-rate of 1.5 Gbps while processing VoIP packets up to a maximum of 600,000 concurrent calls with new calls arriving at 1000 calls/second.

For the attack scenario, we take a 3-minute sample of clean traffic from our trace and merge the call spam attack towards the end of the trace. In particular, one existing caller URI/IP address from our trace is selected as the spammer. We generate multiple SIP requests from this caller towards randomly generated callees. We generate (i) *high volume spam* that lasts a duration of 25 seconds, with new calls generated in the first 10 seconds at the rate of 100 calls/second to yield a total of 1000 spam calls in the trace and (ii) *low volume spam*, consisting of only 10 calls generated by the spammer in the same time duration of 10 seconds. Each spam call is generated to last the same duration of 15 seconds, assuming that the spammer is transmitting the same automated message to each caller. In this scenario, we also assume 100% of the callees respond to the caller and none of the callees hang up on the call before the spam call is completed (terminated by `BYE`). For detecting the attack, we configure the module with a time slot of 2 seconds, where the learning period T_{learn} lasts for 40 time slots. The averaging time periods T_1 and T_2 are 5 and 15 slots respectively.

Fig. 8 (a),(b), and (c) show two of the several features used in detecting the high volume spam attacks (see [5] for figures of other features). Note that each feature can be observed to be stationary before the attack. Observe that there are two peaks in Fig. 8(a), where the first peak occurs close to the beginning of spam and consists of the flood of `INVITE` messages and the second peak occurs close to the end of the spam consisting of `BYE` packets. Our algorithm is able to detect the attack almost instantaneously around time slot 60, when the features of *total requests* reach the *Red* alert stage (alert value = 3). Note also that the histograms (distributions) of callee URI and callee IP addresses are dominated by the spammer and thereby the corresponding UAD exhibits sharp decreases around the beginning of the attack (Fig. 8(b) and (c)). Furthermore, at the time when the alerts for these features turn *Red*, comparing the current histogram with the last clean histogram (the one at time slot 60) would reveal the caller URI and caller IP involved in the attack. The efficacy of our anomaly detection in detecting even the *low volume spam* attacks is highlighted via Fig. 8(d),(e),and (f), which show the performance in the presence of the low volume call spam. In this case, the spammer is able to hide within the background traffic quite well as none of the volume features of *total number of requests* exhibits any significant change during the spam period. However, the intrinsic behavior of the spammer of generating multiple requests from the same client IP address (and user URI) results in the detection of the attack via the features that track user behavior. Hence, the UAD for caller IP addresses exhibits three consecutive alerts, leading to the extraction of the spammer's IP address around time slot 63 (Fig. 8(e) and (f)).



(a) Spam: requests (b) Spam: client IPs (c) Low Spam: requests (d) Low Spam: client IPs

Figure 8. Features used in detecting a VoIP spam initiated via (Call Proxy) Server 1

6 Related Work

While there is a considerable volume of white papers and surveys regarding various vulnerabilities and security threats towards VoIP services (see, e.g., [6]), there is relatively few research studies on analysis of VoIP traffic characteristics. Most focus on defense against specific attacks, e.g., malformed SIP message format attacks [2, 3], DoS and other call disruption attacks [8, 7, 10], and voice spams [1]. However, these studies are not based on real-network SIP traces, and cannot diagnose what cause abnormal problem (as we pinpointed the reason of anomaly in Section 5). To the best of our knowledge our study is the first analysis of SIP traffic from an operational VoIP service and the first attempt at profiling SIP-based VoIP traffic behavior based on real-network traces.

7 Conclusions

In this paper, we have presented a general profiling methodology for characterizing SIP-based VoIP traffic behaviors at multiple levels: the SIP server host, service entity (e.g., registrar and call proxy) and individual user levels. Applying knowledge about application protocol semantics and expected system/user behaviors, an ensemble of statistics and features is selected at each level to capture the essential and stable characteristics of SIP message exchanges, types, volumes, user activities, and so forth. Through our analysis of SIP traffic traces obtained from an operational VoIP service, we show that overall SIP-based VoIP traffic exhibits stable characteristics and behavior that are well captured by the statistics and features selected in our profiling methodology, thereby justifying the selection of these statistics and features. Finally we illustrate how our profiling methodology can be used to help identify anomalies for problem diagnosis and attack detection. In particular, we have developed a novel profiling-based anomaly detection algorithm and demonstrate its efficacy in detecting VoIP attacks through testbed experimentation.

References

[1] R. Dantu and P. Kolan. Detecting spam in VoIP networks. In *Proc. of USENIX, Steps for Reducing Unwanted Traffic on the Internet (SRUTI) Workshop*, pages 31–37, Cambridge, MA, July 2005.

[2] D. Geneiatakis, T. Dagiuklas, C. Lambrinoudakis, G. Kambourakis, and S. Gritzalis. Novel protecting mechanism for SIP-based infrastructure against malformed message attacks: Performance evaluation study. In *Proc. of the 5th International Conference on Communication Systems, Networks and Digital Signal Processing (CSNDSP'06)*, Patras, Greece, July 2006.

[3] D. Geneiatakis, G. Kambourakis, T. Dagiuklas, C. Lambrinoudakis, and S. Gritzalis. SIP message tampering: The SQL code injection attack. In *Proc. IEEE of the 13th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM'05)*, Split, Croatia, Sept. 2005.

[4] A. R. Hickey. For VoIP, too many threats to count. http://searchsecurity.techtarget.com/originalContent/0,289142,sid14-gci1153718,00.html.

[5] H. J. Kang, Z.-L. Zhang, S. Ranjan, and A. Nucci. SIP-based VoIP traffic behavior profiling and its applications. Technical report, NARUS, July 2006.

[6] S. McGann and D. C. Sicker. An analysis of security threats and tools in SIP-based VoIP systems. In *2nd Workshop on Securing Voice over IP*, Washington DC, USA, June 2005.

[7] B. Reynolds and D. Ghosal. Secure IP telephony using multi-layered protection. In *Proc. of Network and Distributed System Security Symposium(NDSS'03)*, San Diego, CA, Feb. 2003.

[8] B. Reynolds, D. Ghosal, C.-N. Chuah, and S. F. Wu. Vulnerability analysis and a security architecture for IP telephony. In *IEEE GlobeCom Workshop on VoIP Security: Challenges and Solutions*, San Diego, CA, Nov. 2004.

[9] J. Rosenberg, H. Schulzrinne, G. Camarillo, P. J. Johnston, A., R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, June 2002.

[10] Y.-S. Wu, S. Bagchi, S. Garg, and N. Singh. SCIDIVE: A stateful and cross protocol intrusion detection architecture for Voice-over-IP environments. In *Proc. of the 2004 International Conference on Dependable Systems and Networks (DSN'04)*, pages 433–442, Split, Croatia, June 2004.