SCIDIVE: A Stateful and Cross Protocol Intrusion Detection Architecture for Voice-over-IP Environments

Yu-Sung Wu, Saurabh Bagchi
School of Electrical & Computer Eng., Purdue University
Email: {yswu,sbagchi}@purdue.edu

Sachin Garg, Navjot Singh, Tim Tsai
Avaya Labs
Email:{sgarg,singh,ttsai}@avaya.com

Abstract

Voice over IP (VoIP) systems are gaining in popularity as the technology for transmitting voice traffic over IP networks. As the popularity of VoIP systems increases, they are being subjected to different kinds of intrusions some of which are specific to such systems and some follow a general pattern. VoIP systems pose several new challenges to Intrusion Detection System (IDS) designers. First, these systems employ multiple protocols for call management (e.g., SIP) and data delivery (e.g., RTP). Second, the systems are distributed in nature and employ distributed clients, servers and proxies. Third, the attacks to such systems span a large class, from denial of service to billing fraud attacks. Finally, the systems are heterogeneous and typically under several different administrative domains.

In this paper, we propose the design of an intrusion detection system targeted to VoIP systems, called SCIDIVE (pronounced "*Skydive*"). SCIDIVE is structured to detect different classes of intrusions, including, masquerading, denial of service, and media stream-based attacks. It can be installed at multiple points – clients, servers, or proxies, and can operate with both classes of protocols that compose VoIP systems – call management protocols (*CMP*), e.g., the Session Initiation Protocol (SIP), and media delivery protocols (*MDP*), e.g., the Real Time Transport Protocol (RTP). SCIDIVE proposes two abstractions for VoIP IDS – *Stateful detection* and *Cross-protocol detection*. Stateful detection denotes the functionality of assembling state from multiple packets and using the aggregated state in the rule matching engine. Cross protocol detection denotes the functionality of matching rules that span multiple protocols, e.g., detecting a pattern in a SIP packet followed by one in a succeeding RTP packet followed by one in a H.323 packet. SCIDIVE is demonstrated on a sample VoIP system that comprises SIP clients and SIP proxy servers. Four attack scenarios are created and the accuracy and the efficiency of the system evaluated with rules meant to catch these attacks.

Keywords: Intrusion detection, Voice over IP system, Cross-protocol detection, Stateful detection, SIP, RTP.

1 Introduction

Voice over IP (VoIP) systems are gaining in popularity as the technology for transmitting voice traffic over IP networks. While VoIP technology is set to revolutionize communications, and is already being used by a number of traditional telephone companies to connect their regional offices, on a smaller scale it can also be a useful solution for businesses looking to trim their telephone expenses. As the popularity of VoIP systems increases, they are being subjected to different kinds of intrusions, some of which are specific to such systems, and some of which follow a general pattern. There have been enormous strides made in the field of intrusion detection systems (IDS) for different components of the information technology infrastructure. Some of the IDSs are generic in nature and can be customized with detection rules specific to the environment in which they are deployed (e.g., Snort [13] and Prelude [17]), and some are tools specifically targeted to an environment or to specific classes of intrusions, such as IBM Tivoli Intrusion Manager for MQSeries products [18]. VoIP systems pose several new challenges to IDS designers. First, these systems employ multiple protocols for call management and data delivery. Second, the systems are distributed in nature and employ distributed clients, servers, and proxies. Third, the attacks against such systems span a large class, from denial of service to billing fraud. Finally, the systems are heterogeneous and typically under several different administrative domains, e.g., the proxy server may be provided by the service provider and the client managed by the home organization.

In this paper, we propose the design of an intrusion detection system targeted to VoIP systems, called SCIDIVE (pronounced "*Skydive*"). SCIDIVE is structured to detect different classes of intrusions, including, masquerading, denial of service, and media stream-based attacks. It can be installed at multiple points – clients, servers, or proxies, and can, without substantial system customization, be extended for detecting new classes of attacks. The IDS can handle client mobility, an important design goal of VoIP protocols such as SIP, and does not flag false alarms for such situations. SCIDIVE can operate with both classes of protocols that compose VoIP systems – call management protocols (*CMP*), e.g., the Session Initiation Protocol (SIP), and media delivery protocols (*MDP*), e.g., the Real Time Transport Protocol (RTP).

SCIDIVE proposes two abstractions for VoIP IDS – *stateful detection* and *cross-protocol detection*. Stateful detection denotes the functionality of assembling state from multiple packets and using the aggregated state in the rule-matching engine. The reassembly functionality is applicable to packets of both CMP and MDP and can be configured to handle packets spread out arbitrarily far apart in time. Some existing IDSs provide support for reassembly, but they are restrictive and applicable only to specific protocols. For example, Snort's *stream4* module can reassemble TCP packets that belong to the same session. Cross protocol detection denotes the functionality of matching rules that span multiple protocols, e.g., detecting a pattern in a SIP packet followed by one in a succeeding RTP packet followed by one in an RTCP packet. The aggregation across protocols can be chained in an arbitrarily long manner and spread out in time. This abstraction is powerful for VoIP systems because they involve multiple protocols and several attacks are based on sequences that cross protocol boundaries. There are very few systems today that support cross-protocol detection. One of the notable ones is WebSTAT [1] for detecting attacks against web servers by correlating protocols in a vertical stack, e.g., application level (web server) and operating system log. Since VoIP systems use multiple application layer protocols, horizontal cross-protocol correlation is required.

The architecture of SCIDIVE uses a *Distiller*, through which all incoming network traffic passes and which translates packets into protocol dependent information units called *Footprints*. The Footprints that belong to the same session are grouped into *Trails*. The *Event Generator* maps Footprints into *Events* which are matched by the *Rule Matching Engine* against a *Ruleset*. According to the stateful and cross-protocol philosophies, the Events can potentially have state information and encapsulate information from multiple packets.

SCIDIVE is demonstrated on a sample VoIP system that comprises SIP clients and SIP proxy servers. The system uses SIP Express Router for the proxy and three different kinds of clients – KDE's KPhone [2], Microsoft Windows Messenger [3], and XTen's X-Lite IP Telephony client [4]. The protocols used are SIP for call management and RTP for real-time audio data transfer. In our experiments, an instance of SCIDIVE is associated with each client. Four different types of attacks are simulated on the system and the effectiveness and efficiency of SCIDIVE analyzed.

The rest of the paper is organized as follows. Section 2 gives an overview of VoIP systems and attacks in such systems. Section 3 presents the architecture of SCIDIVE and motivates the stateful and cross-protocol detection through two running examples. Section 4 presents the implementation of SCIDIVE, the kinds of attacks simulated in the system, and analysis of the efficiency of the system. Section 5 reviews related work. Section 6 concludes the paper.

2 System Description: VoIP Systems and Attack Classification

2.1 VoIP Overview

Voice over IP (VoIP) systems provide facilities for setting up and managing voice communications based on one of two main protocols: H.323 [5] and SIP [6]. H.323 is the most widely deployed standard in VoIP communications, but SIP is increasing in popularity due to its simplicity and corresponding ease of implementation. With both types of system, endpoints or terminals, which may be physical phones (hardphones) or software programs executing on a general-purpose computer (softphones), send and receive RTP [7] packets that contain encoded voice conversations. Since voice calls may be made between IP phones and phones on the Public-Switched Telephone Network (PSTN), gateways often perform transparent translation between IP and non-IP based networks. Such gateways may implement protocols for media gateway management such as MGCP [7] and MEGACO/H.248 [9].

Within an H.323 network, an optional gatekeeper may be present. The gatekeeper performs several functions including authorizing network access, assisting in managing quality of service, and providing address-translation services. Also, multipoint controllers may be present to manage multipoint conferences between three or more terminals or gateways.

SIP networks also include additional types of servers. A proxy server forwards requests, possibly after performing some processing or translation. A redirect server is used to support mobile clients and performs address translation for an accepted request and returns the new address to the originator of the request. Both proxy and redirect servers may be used to implement call forwarding and other similar services. User agent clients send requests to user agent servers to initiate calls. The user notifies a registrar of his current location to allow others to contact him. The registrar is often combined with a proxy or redirect server.

Both H.323 and SIP provide protocols for call setup, management, and media delivery. Voice is encoded using a negotiated codec and delivered using RTP over UDP/IP for both protocols. However, call setup and management are handled quite differently. H.323 relies on the H.225.0 [10] and H.245 protocols [11], whereas SIP uses a much simpler set of request messages: INVITE, ACK, OPTIONS, BYE, CANCEL, and REGISTER. SIP provides a globally reachable address to which callees bind using SIP REGISTER method. The INVITE message is used by a user client agent wishing to initiate a session, which can be responded to with an OK, followed by an ACK. To tear down a connection, a BYE message is sent. CANCEL cancels a pending invite. OPTIONS is used to query or change optional parameters of the session, such as encryption.

Figure 1 illustrates a typical set of SIP messages leading to a session.

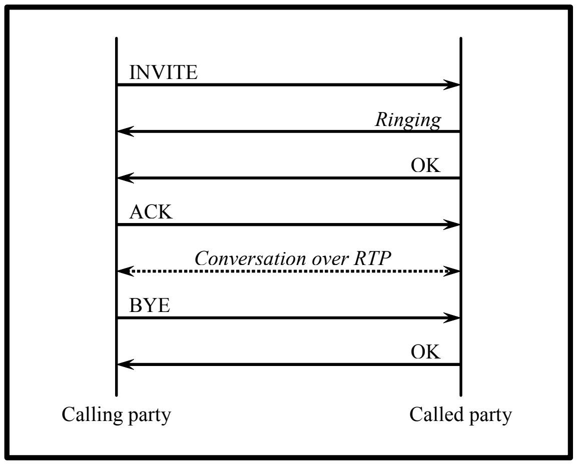


Figure 1: Sample SIP message exchange in a SIP call setup and teardown

2.2 VoIP Attacks

One of the main advantages of a VoIP system is the convergence of voice and data networks with voice being conveyed over a data network. While this offers advantages in cost and ease of management, the use of the data network in a converged system makes the voice network vulnerable to the same vulnerabilities suffered by the data network. This includes well-known attacks such as denial of service attacks as well as authentication attacks. In addition, a voice network introduces potential vulnerabilities related to toll fraud, privacy, and denial of service attacks based on degrading the quality of service of the voice conversation.

A major source of vulnerabilities lies in the protocols used to set up and manage calls. Both H.323 and SIP transmit packet headers and payload in clear text, which allows an attacker to forge packets that manipulate device and call states. For example, such forged packets can prematurely terminate calls, redirect calls, or facilitate toll fraud. Some efforts are currently underway to develop encrypted signaling, but no solution has found widespread adoption.

In addition to vulnerabilities present in the signaling protocols, the RTP protocol for media delivery also introduces several vulnerabilities due to the absence of authentication and encryption. Each RTP packet header contains a sequence number that allows the recipient to play back voice packets in the proper order. However, an attacker can easily inject artificial packets with higher sequence numbers that will cause the injected packets to be played in place of the real packets. An attack can also fake the SSRC field to impersonate another participant in a call.

3 SCIDIVE Architecture

3.1 SCIDIVE Components: Footprints, Trails, Events, Rules

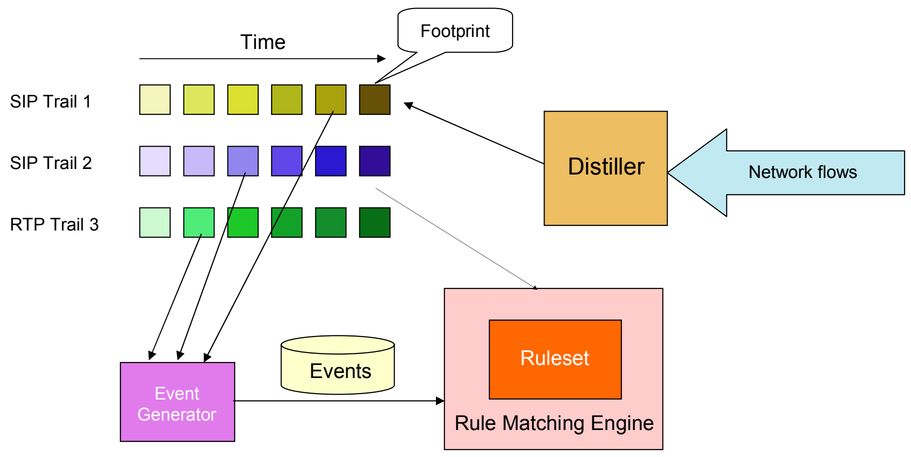


Figure 2: Overview of SCIDIVE Components

Figure 2 presents an overview of the SCIDIVE architecture. In SCIDIVE, incoming network flows first pass through the ***Distiller***, which translates packets into protocol dependent information units called ***Footprints***. The Distiller is responsible for doing IP fragmentation, reassembly, decoding protocols, and finally generating the corresponding Footprints. A ***Footprint*** is a protocol dependent information unit, which, for example, could be composed of a SIP message or an RTP packet. Footprints that belong to the same session are grouped into ***Trails***. In Figure 2 we have three Trails that correspond to two SIP sessions and one RTP session.

The ***Event Generator*** maps footprints into a single event. For example**,** we can map two out of order RTP Footprints into an event called *RtpJitter*. Event Generator is hard-coded and seamlessly coupled with internal structures for best possible performance. In general, it is just a layer of abstraction, which correlates the information in footprints and concentrates the information into a single event. It helps performance by hiding some computationally expensive matching, e.g., by triggering the ruleset at the moment of interest instead of triggering it upon each incoming RTP Footprint.

Ruleset is triggered by a sequence of Events, e.g., we can define a rule for detecting RTP flow [event 1] after a session is torn down [event 2]. The matching in the Ruleset is based on Events that can potentially encapsulate information from multiple packets and can bear state information. Besides the information that Events provide, the Ruleset can also perform the matching based on crude information directly from the Trails in case no suitable Event is available. For example, we might be interested in knowing who prematurely tears down the session. To achieve this, we probably need to have a look at the corresponding SIP Footprint to identify the ID and IP address of the originator. This direct access however will cause some degree of inefficiency.

3.2 Cross-protocol Methodology for Detection

We propose a powerful abstraction for intrusion detection systems in general, and VoIP IDSs in particular, namely, cross-protocol detection. An IDS that uses cross-protocol detection accesses packets from multiple protocols in a system to perform its detection. This methodology is suitable to systems that use multiple protocols and where attacks spanning these multiple protocols are possible. There is the important design consideration that such access to information across protocols must be made efficiently.

A VoIP system incorporates multiple protocols. A typical example is the use of SIP to establish a connection, followed by use of RTP to transfer voice data. Also, RTCP and ICMP are used to monitor the health of the connection. VoIP systems typically have application level software for billing purposes and therefore may have accounting software and a database.

To motivate the need for cross-protocol detection, we introduce a synthetic example of a billing fraud attack. Since VoIP systems have been gaining in popularity only of late, there are very few instances of actual attacks in databases such as CERT [15] and Bugtraq [16]. In our synthetic scenario, the attack is launched by the attacker exploiting a vulnerability in the SIP proxy. She sends a carefully crafted SIP message to fool the proxy into believing the call is initiated by someone else. The proxy initiates the accounting software with the information about the incorrect source for the call. This allows the attacker to make calls without being charged.

Using the cross-protocol methodology for detection, one can create a cross-protocol rule to look at the SIP messages, the transaction messages between the accounting software and the database, and the RTP flows later on. Specifically, each of the following three conditions must hold.

1. The SIP message should follow the correct format.
2. When the accounting software sends out a transaction to denote a call from user A to user B, check if user A has sent a SIP Call Initialization message to user B. If user A has not set up the call with a legitimate SIP Call Initialization message, then this condition will be violated.
3. Check the source/destination IP addresses of the subsequent RTP flows. Together with information from DNS and SIP Location Servers, we can reconfirm that each RTP flow has a corresponding legitimate call setup.

In SCIDIVE, cross-protocol detection is achieved through keeping multiple trails for different sessions. In our example, we can have (i) a 'SIP trail' which tracks all the SIP messages in the session between user A and user B; (ii) an 'RTP trail' which tracks all the RTP packets in the session between A and B; and (iii) an 'Accounting trail' which tracks relevant accounting transactions in this session between A and B. Then, we can define three events based on the three trails corresponding to the three conditions above. The first event is "an incorrectly formatted SIP message in the SIP trail", which could be an indication of an attempt to exploit the vulnerability in the SIP proxy. The second event is "a transaction in the Accounting trail that has no matching call initialization message in the SIP trail". The third event is "either the source or destination IP addresses of the RTP packet without a matching address in the SIP packet". The third event is specialized to take mobility into account, which will be indicated by a SIP REINVITE message with an update of state at the SIP Registrar that maintains location information. In the Ruleset, we can put a rule called *Billing Fraud,* which is triggered by a combination of these three events. An advantage of creating a rule based on a sequence of three events is improving the accuracy of the alarm because the rule is based on three facets of the attack. It is perceivable that relying solely on Event 1 or Event 3 to signal 'Billing Fraud' alarm will result in false alarms. Also, bugs or temporary system failures might cause Event 2. Therefore, relying solely on Event 2 will possibly give us false alarms.

3.3 Stateful Methodology for Detection

A second abstraction useful for VoIP systems in particular is stateful detection. Stateful detection implies building up *relevant* state *within* a session and *across* sessions and using the state in matching for possible attacks. It is important that the state aggregation be done efficiently so that the technique is applicable in high throughput systems, such as VoIP systems.

A VoIP system maintains considerable amount of system state. The client side maintains state about all the active connections – when the connection was initiated, when it can be torn down, and what the properties of the connection are. The server side also maintains state relevant to billing, such as the duration of the call.

To motivate the need for stateful detection, we introduce a synthetic example of a DoS attack and a password guessing attack. An unauthorized user client keeps sending unauthenticated REGISTER requests to bombard the SIP proxy and ignores the 401 UNAUTHORIZED reply error message from the SIP proxy. If the user client keeps sending the same request to the server, it can be seen as a type of DoS attack on the SIP proxy. Along with the UNAUTHORIZED reply message, the proxy sends a challenge phrase to the client. If the client keeps sending requests with different values in the challenge response field, this could be seen as a type of attack that is trying to break the authentication key by brute force. In either case, it would be helpful for detection if the system can look at the series of user client requests and the subsequent server responses. Since 4XX responses are not uncommon in a normal session, a traditional IDS like Snort with a rule to detect multiple 4XX responses may flag a large number of false alarms. For example, most user clients send an unauthenticated REGISTER request to the server, presuming that the SIP Proxy does not require authentication.

Later, the server sends a 401 response along with a challenge phrase to the client to indicate that authentication is required. The client should then send a new REGISTER request to the server along with the correct response phrase to continue the registration process. If the IDS does not isolate 4XX error messages from different sessions and doesn't correlate 4XX error messages with requests, it is likely it will make false verdicts based on unrelated 4XX error messages.

In SCIDIVE, Footprints that belong to a session are structured and kept in a single trail. Therefore, the history of all the state transitions of each session can be easily tracked. To handle the two attack scenarios above, we can set up the following two events – (i) an event "DoS via repeated SIP requests", which represents continuous, alternating SIP requests and 4XX error messages in a particular session; (ii) an event "Password guessing" which represents continuous, alternating SIP requests with different challenge responses and 401 Unauthorized reply error messages in a particular session. Flagging of the two events indicates two different kinds of attacks that may have different responses. In the first case, the response may be to identify the source of the attacker and block her IP address while in the second case, it may be important to take more stringent measures to ensure the security of the system, such as changing the authentication password to a longer one.

3.4 Placement of SCIDIVE Components

The SCIDIVE architecture has a great deal of flexibility in terms of the placement of its components. For example, it is possible to deploy the SCIDIVE IDS only on the SIP client side for detecting anomalies in the VoIP traffic in and out of the client. Also, we have shown in previous work [12] that doing correlation on alerts from multiple detectors could increase the detection accuracy. We can use a similar idea by deploying SCIDIVE-enabled IDS on both end-points of the VoIP system, i.e., both clients. In such an installation, the two IDSs could exchange event objects and portions of trails to enhance the overall detection accuracy and efficiency. A typical example of the benefits of using IDS collaboratively in VoIP system is verifying the integrity of the exchanged messages, so that spoofed traffic from an attacker can be detected. Yet, a more aggressive approach would be to deploy the SCIDIVE IDS on all the components – Clients, SIP Proxy, and Registrar server – in a VoIP system. For example, in the Billing attack scenario outlined in Section 3.2, we need to deploy the IDS on the SIP Proxy and the Accounting Server to detect Event 1 and Event 2. Also we need to deploy the IDS close to both clients to monitor RTP flows to detect Event 3. While it is pretty obvious that using multiple SCIDIVE IDS and an alert correlation engine will give better detection accuracy, the actual gains, impacts on the system performance, and costs for doing so remain an interesting research subject that we will investigate in the future.

4 Prototype and Experiments

An IDS prototype is built to instantiate the SCIDIVE architecture for VoIP systems. We implement four attacks against the VoIP system, instantiate rules in SCIDIVE for detecting the attacks, and perform analysis of the detection efficiency. For simplicity, the IDS is placed at the client end for the experiments in this paper. This configuration is shown in Figure 3 and is referred to as an *End-point based IDS architecture*. In this architecture, an IDS instance is associated with each client.

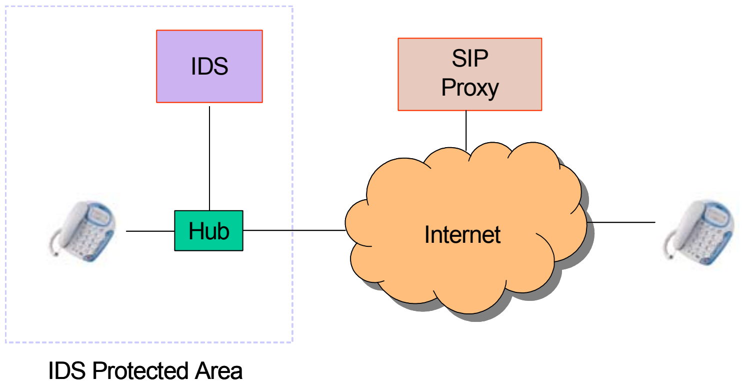


Figure 3: IDS Engine sits on or close to the end-point

4.1 Testbed

Our testbed comprises the following:

- Proxy : SIP Express Router from www.iptel.org
- Clients : Kphone from www.wirlab.net; Windows Messenger from Microsoft; X-Lite from www.xten.com

Figure 4 depicts the topology of our testbed. The IDS is connected to a hub through which the traffic of Client A can be seen. Although the prototype IDS can also see the traffic of Client B and the SIP Proxy, it does not look into those traffic for any purpose, thus mimicking an end-point based IDS.

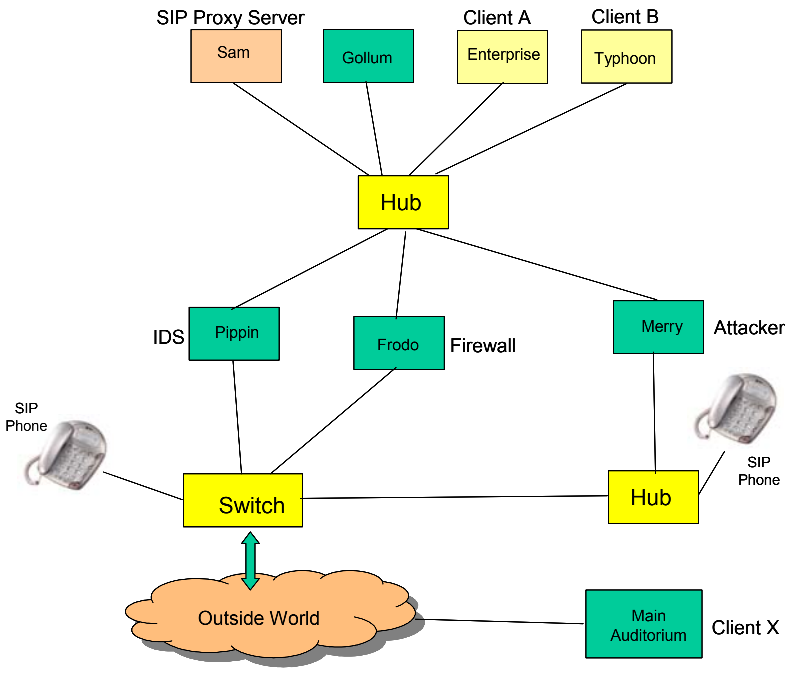


Figure 4: SCIDIVE testbed with client phones, proxy servers, network elements, attacker host and the IDS

4.2 Attacks and Demonstration of the IDS

We implement four attacks to demonstrate the functionality of the IDS. Three of the four attacks are based on the vulnerabilities in the signaling protocol, i.e., SIP. They are BYE attack, Fake Instant Messaging, and Call Hijacking. The fourth is based on the vulnerabilities in the media transport protocol, which is the RTP attack. A summary of the attacks is presented in Table 1.

Name of attack	**Protocols involved**	**Cross-protocol or not ? If yes, how ?**	**Stateful or not? If yes, how ?**	**Rule snippet**
Bye attack	SIP, RTP	Yes. Detect no RTP traffic once SIP BYE has been seen.	Yes. Monitor the session to determine when a session has been torn down.	No RTP traffic should be seen after a SIP BYE from a particular user agent.
Fake Instant Messaging	SIP, IP	Yes. Check the source IP addresses of incoming IM messages (SIP Message).	No.	Check the IP addresses of all the incoming messages. The IP address should stay the same within a time period.
Call Hijacking	SIP, RTP	Yes. Detect no RTP	Yes. Monitor the	No RTP traffic

		traffic once SIP REINVITE has been seen.	session to determine when a session has been redirected.	should be seen after a SIP BYE from a particular user agent.
RTP Attack	RTP, IP	Yes. Check the source IP address of the RTP packets.	Yes. Check if the sequence numbers in consecutive packets increase expectedly.	Check if RTP packets come from legitimate IP address and if the sequence number increases appropriately.

Table 1: Summary of Attacks

Details of the attacks are presented in the following sections.

4.2.1 BYE Attack

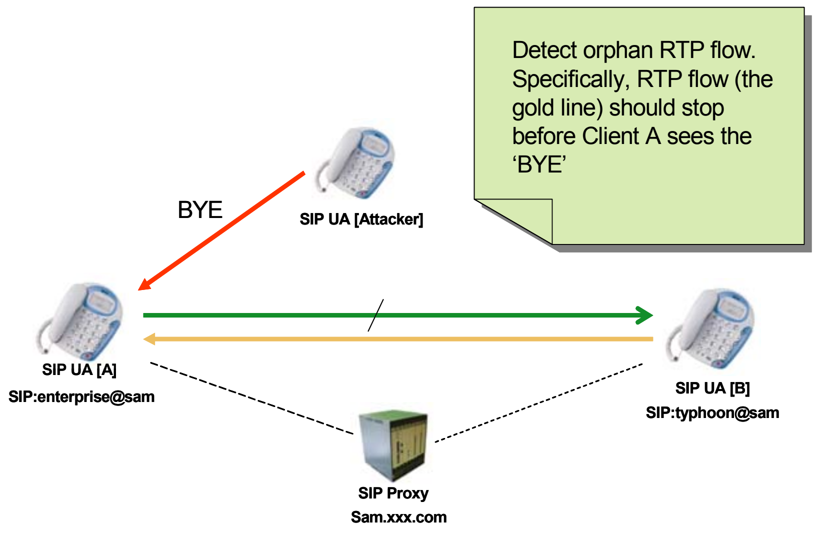

Detect orphan RTP flow. Specifically, RTP flow (the gold line) should stop before Client A sees the 'BYE'
BYE
SIP UA [Attacker]
SIP UA [A]
SIP:enterprise@sam
SIP UA [B]
SIP:typhoon@sam
SIP Proxy
Sam.xxx.com

Figure 5: Schematic of BYE Attack

In this attack scenario, we have three SIP User Agents {A, B, and Attacker}. We also have a SIP Proxy for setting up the connections. The goal of the BYE attack is to prematurely tear down an existing dialog session, which can be viewed as a kind of Denial-of-Service attack. In the above figure, the gold and green lines going respectively from SIP UA B to SIP UA A, and vice-versa, represent an ongoing dialog session between A and B. Later, Attacker sends a faked BYE message to A. After that, A will believe that it is B who wants to tear down the connection by sending the BYE message. A will stop its outward RTP flow immediately, while B will continue to send RTP packets to A, since B has no notion that the connection should be terminated.

In order to detect this attack, we create a rule that detects orphan RTP flow. Specifically, if it is indeed B who wants to stop the connection, then A should not see the RTP flow from B after getting the BYE message. Thus, in the IDS, we create a rule which signals an alarm when seeing new RTP Footprints in the RTP Trail that corresponds to the flow from B after seeing a BYE SIP Footprint. Although the attack itself occurs only within the signaling protocol (SIP), our detection rule spans SIP and RTP and provides an illustration of the importance of cross-protocol detection.

4.2.2 Fake Instant Messaging

In addition to IP Phone Call setup, SIP also supports Instant Messaging. By faking the header of an instant message appropriately, the attacker can forge a message to A and mislead it into believing the message is from B.

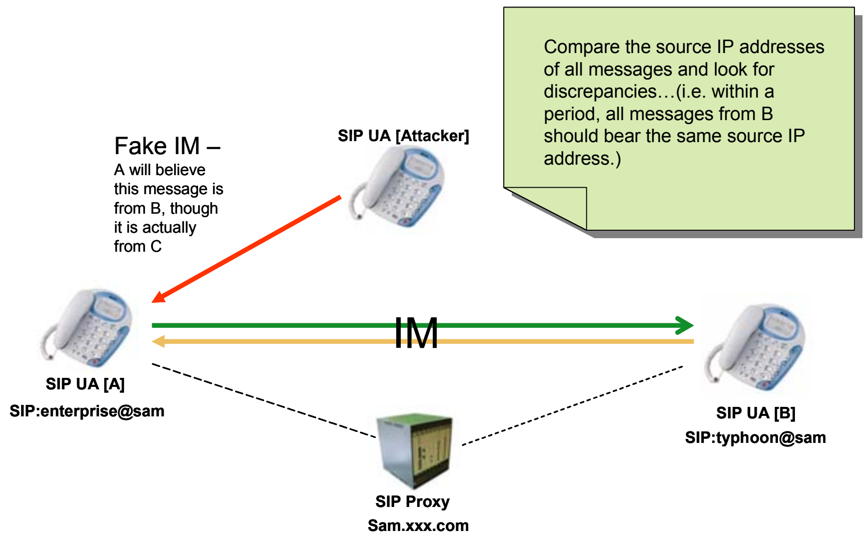

Compare the source IP addresses of all messages and look for discrepancies…(i.e. within a period, all messages from B should bear the same source IP address.)
Fake IM –
A will believe this message is from B, though it is actually from C
SIP UA [Attacker]
IM
SIP UA [A]
SIP:enterprise@sam
SIP UA [B]
SIP:typhoon@sam
SIP Proxy
Sam.xxx.com

Figure 6: Schematic of Fake Instant Messaging Attack

Our rule for detecting this attack looks at the IP addresses of the messages. Under most circumstances, within a period, messages from B should bear the same source IP address. Therefore, once we find a message from B which carries a different IP address, it would be an indication that this message is a fake one. The rule takes rate of user mobility into account and allows for changes in the IP address according to the maximum rate of user motion. Indeed, this rule is not perfect. If the attacker is able to spoof its IP address, then this rule will not work. However, based on the Host-based architecture, this is probably the best we can do. This leads to interests in research on a more ambitious architecture like deploying IDS on both client ends.

4.2.3 Call Hijacking

Call Hijacking is also a signaling based attack. In this attack, by sending a REINVITE message to B, the attacker can redirect the RTP flow that is supposed to go to B to another location, most likely the IP address of the machine where the attacker is. Normally, a REINVITE message is used for call migrating. For example, B wants to go outdoors, so it moves the phone call from the landline phone to its cell phone. In this attack, the attacker abuses this feature by sending a fake REINVITE message to fool A into believing that B is going to change its IP address to a new address.

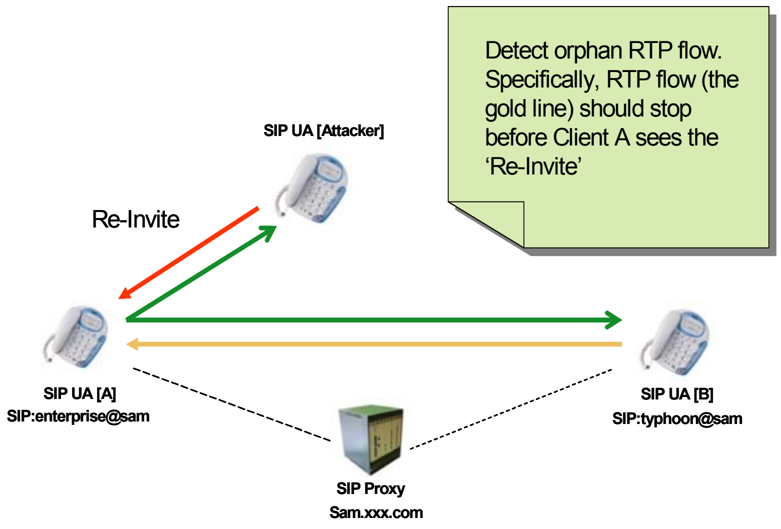

Detect orphan RTP flow. Specifically, RTP flow (the gold line) should stop before Client A sees the 'Re-Invite'
SIP UA [Attacker]
Re-Invite
SIP UA [A]
SIP:enterprise@sam
SIP UA [B]
SIP:typhoon@sam
SIP Proxy
Sam.xxx.com

Figure 7: Schematic of Call Hijacking Attack

A direct impact of this attack is that B will experience a continued silence since A is no longer sending its voice data to B. For this part, the Call Hijacking attack can be seen as a kind of Denial-of-Service attack. A more subtle impact is that if the attacker were able to emulate the voice data of B, then after successfully redirecting the call, A would not be able to distinguish between B and the attacker. This could lead to issues of confidentiality and breach of privacy since the attacker will be able to listen to what A is saying.

To detect this attack, we use a similar approach as for the BYE attack. Intuitively, if the REINVITE message is indeed from B, then A should not see any RTP flow from B after that. By detecting orphan flows, we are able to determine whether it's a legitimate REINVITE message or a malicious one.

4.2.4 RTP Attack

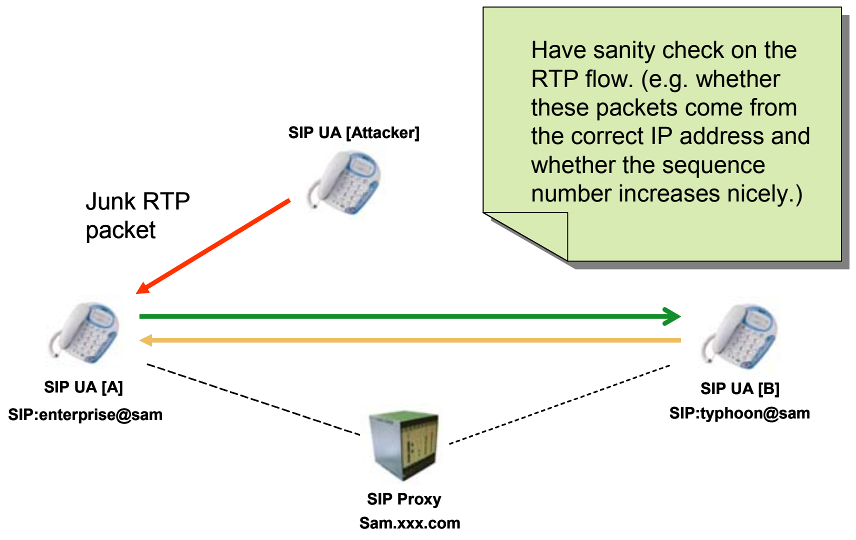

Have sanity check on the RTP flow. (e.g. whether these packets come from the correct IP address and whether the sequence number increases nicely.)
SIP UA [Attacker]
Junk RTP packet
SIP UA [A]
SIP:enterprise@sam
SIP UA [B]
SIP:typhoon@sam
SIP Proxy
Sam.xxx.com

Figure 8: Schematic of RTP Attack

All the attacks mentioned earlier are signaling based attacks. The RTP attack on the contrary is based on the vulnerabilities in media transport. In this attack, the attacker sends RTP packets whose contents are garbage (both the header and the payload are filled with random bytes) to one of the persons in a dialog. In this example, the attacker sends garbage bytes to A.

Since these garbage packets will corrupt the jitter buffer in the IP Phone client, depending on different implementations, this attack could result in intermittent voice conversation or in crashing the client. For example, in our experiment, X-Lite will crash after this attack while the effect on Microsoft Messenger is intermittent voice conversation. This attack also leads to degradation in QoS (jitter) and in the voice quality.

The rule we use for detecting this attack is that the sequence number in succeeding RTP packets should increase regularly. Specifically, if we see two consecutive packets whose sequence numbers have a difference greater than 100, the IDS will signal an alarm.

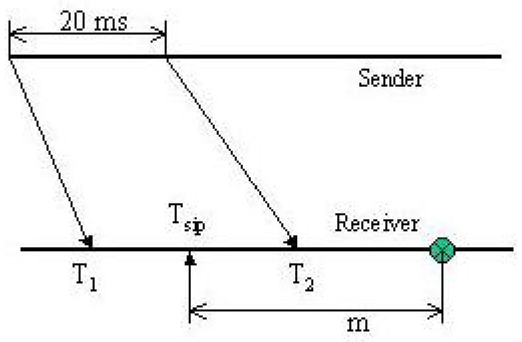
4.3 Performance Evaluation

In this section, we comment on the performance of the IDS system with respect to three metrics: (1) the detection delay, D, which is defined as the time from when an attack/intrusion is made to the time it is detected, (2) the probability of false alarm, P_f, which is defined as the probability that the IDS flags an intrusion when none has occurred, and (3) the probability of missed alarm, P_m, which is defined as the probability that the IDS system does not flag an intrusion when one occurs. The goal is to make the reader aware of the variables that affect these metrics and give an idea of practical values. Detailed performance evaluation with numerical computation is the subject of ongoing work.

4.3.1 *BYE and Call Hijacking attack*

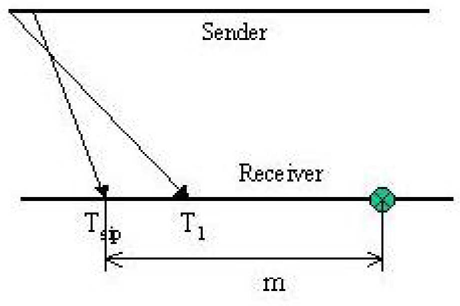
In both these attacks, the IDS rule looks at the SIP signaling event (BYE or REINVITE) and monitors the media stream following this event to detect an intrusion. Specifically, the arrival of an RTP packet at the original RTP port from the original sender flags an intrusion.

- *Detection Delay*: Measuring from the time the SIP message is received, it includes the time up to the arrival of the RTP packet from the original sender. Obviously, the time depends on the frequency of RTP packets. A typical period employed is 20 milliseconds. However, the RTP packet arrival depends also on the network conditions. Specifically, the delay distribution of packets from the sender to the receiver would cause this quantity to change. The above figure outlines the various timing variables involved. Let the time of the last RTP packet arrival be T_1 (before the fake BYE/REINVITE message arrival). Also, without loss of generality,



let this message originate from the sender at time 0. Then T_1 is the transport delay of this RTP packet. Further, let T_{sip} be the time of arrival of the SIP message. Let T_2 be the time of arrival of the RTP packet, which originated 20 ms after the previous RTP packet. We assume that the fake SIP message was generated after the 1st RTP packet but before the 2nd RTP packet. When the SIP message is received, the IDS system starts looking for RTP packets for a total duration of "m" milliseconds. Obviously, T_1, T_2 and T_{sip} are random variables and m is a fixed value. The detection delay D is a function of these four parameters as $D = T_2 - T_{sip}$, where $T_2 = 20 + N_{rtp}$ and $T_{sip} = G_{sip} + N_{sip}$. N_{rtp} and N_{sip} are the random variables associated with the network delay for each packet. While the second RTP packet is generated 20 milliseconds after the origin, the fake SIP message is generated between the two RTP packets with a distribution given by G_{sip}. Therefore, $D = 20 + N_{rtp} - (G_{sip} - N_{sip})$. Given the distributions of these random variables, it is possible to compute the detection delay distribution. Under the simplest of assumptions, where the fake SIP message is generated with a uniform distribution in (0,20), and the network delay is assumed to be independent and identical for all packets, the expected detection delay is 10 milliseconds, which is half of the RTP packet generation period.

- *Probability of Missed Alarm*: Since the detection depends on monitoring after a SIP message arrival and since this monitoring interval is necessary finite (m), there is a probability that the IDS system may not detect the intrusion. For instance, if the subsequent RTP packet(s) from the original sender are lost (somewhere in the network) and no RTP packet arrives within the monitoring interval, then no alarm will be raised. Referring to the above figure, the probability is given by $P_f = \Pr\{T_2 > T_{sip} + m\} = \Pr\{N_{rtp} - G_{sip} - N_{sip} > m - 20\}$.
- *Probability of False Alarm*: Although rare, it is possible for a valid BYE message to arrive before the RTP packet if, for instance, they take a different route, as shown in the following figure. In this case, the IDS system will raise a false alarm. In order to compute this probability, we assume that the sender generated the valid SIP BYE/REINVITE immediately (with zero delay) after it has sent out the last RTP packet. In that case, the false alarm probability is given as $P_f = \Pr\{N_{sip} < N_{rtp}\}$. If the density function and distribution function of N_{rtp} and N_{sip} is assumed to be identical and independent denoted by $f_N(t)$ and $F_N(t)$ respectively, then $\int_0^m F_N(t) f_N(t)dt$.



5 Related Work

Currently, we are not aware of any IDS dedicated to VoIP systems. One possibility is to use a general purpose network IDS like Snort [13] for the purpose. Such a network based IDS would sniff the packets arriving at a host and attempt to find predefined patterns indicating an attack in the packets. For example, to detect the OpenSSL worm traffic, Snort searches each packet that flows through port 443 for the pattern 'TERM=xterm', which indicates the initialization of a UNIX terminal and is part of the behavior of the OpenSSL worm. One potential problem of this approach is that if the target pattern is fragmented across multiple packets, then the IDS will miss it. Seeing this problem, Snort has an IP fragmentation-reassembly module which assembles fragmented IP packets. Also, for TCP packets, it has a *stream4* reassembly module that can aggregate TCP packets within the same TCP session (like a FTP session) into a conglomerate pseudo packet. After this, the same pattern-matching algorithm is employed on the pseudo packet.

The matching infrastructure provided by Snort is versatile and has been proven to be very effective for dealing with most network based intrusions. However, for VoIP applications, there are two drawbacks to using Snort directly:

1. No reassembly functionality is available for grouping UDP packets that belong to a VoIP session. This means malicious patterns that are spread across UDP packets would elude the matching rule in Snort.
2. Snort's detection is rather session unaware. It does provide stateful detection for some TCP applications like HTTP and FTP. However, there are currently no infrastructures that can help in processing stateful VoIP sessions. This could pose a problem for the detection accuracy and efficiency of detection.

The WebSTAT system [1] provides stateful intrusion detection for web servers. It builds on STAT [14] which supports the modeling of multi-step, complex attacks in terms of states and transitions. WebSTAT operates on multiple event streams, and is able to correlate both network-level and operating system-level events with entries contained in server logs. WebSTAT uses some language extensions specific to web server attacks, and event generators that can read web server log, parse them and create events in a commonly understandable form. However, the work is essentially an alert correlation engine and does not show evidence of using considerable state across protocols. Also, the state is gathered from vertically layered elements in the protocol stack which operate sequentially (e.g. a web server and the OS), while SCIDIVE performs state aggregation across concurrently executing application level protocols.

6 Conclusions

In this paper we have presented the design and implementation of an intrusion detection system called SCIDIVE for protecting VoIP systems. The protected system uses multiple application protocols for signaling and data transport, of which SIP and RTP are used respectively for the demonstration. SCIDIVE introduces two important abstractions for detection in VoIP systems – stateful detection and cross-protocol detection. In the former, state can be assembled from multiple packets and the aggregated state can be used in the rule-matching engine. Cross protocol detection denotes the functionality of matching rules that span multiple protocols. The implementation of SCIDIVE uses the two abstractions to detect different kinds of attacks. The capability of the system is demonstrated through four kinds of attacks – three of which are a mix of signaling and data transport attacks (Call hijacking, Fake instant messaging, and Bye attacks) and one is a data transport based attack (RTP attack). The performance of the IDS with respect to detection delay, probability of false alarm, and probability of missed alarm are analyzed.

In the future, we plan to investigate cooperative detection between multiple SCIDIVE components. We plan to investigate where the components should be placed and what kinds of state needs to be exchanged between them. A challenge is to design the appropriate protocol that does not overwhelm the system with control messages from the detectors. This may necessitate a hierarchical decomposition of the system with different layers looking at different levels of abstraction for the system. For example, one can consider a SCIDIVE detector for SIP clients in each LAN, while a higher level detector is responsible for monitoring clients in an entire WAN comprising multiple LANs. We plan to evaluate the effectiveness and accuracy of SCIDIVE through simulated attacks. This is a difficult enough task for systems with widely known exploits. It is even more so for VoIP systems that do not have widely publicized attack scenarios. We are in the process of collecting sample attacks that have been seen by a VoIP system vendor and developing attack scripts to mimic them. We anticipate that the accuracy of the detection will be a function of the input rule base as well as the design of the SCIDIVE components. The efficiency of the algorithm for creating events from footprints and matching events against the rule set will affect the detection latency in addition to the structure of the rules.

References

[1] Giovanni Vigna, William Robertson, Vishal Kher, Richard A. Kemmerer, “A Stateful Intrusion Detection System for World-Wide Web Servers,” To Appear in Proceedings of the 19th Annual Computer Security Applications Conference, December 8-12, 2003, Las Vegas, Nevada.

[2] Debian GNU/Linux, “KDE K-Phone,” Available at: http://www.wirlab.net/kphone/

[3] Microsoft, “MSN Messenger v. 6.1,” Available at: http://messenger.msn.com/

[4] X-Ten Networks, “X-Lite IP Telephony Client,” Available at: http://www.xten.com/index.php?menu=products&smenu=xlite

[5] ITU-T, “Packet-based multimedia communications systems,” Recommendation H.323, February 1998.

[6] M. Handley *et. al.*,“SIP: Session Initiation Protocol,” RFC 2543, March 1999.

[7] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF, RFC 3550, July 2003,Available at: http://www.ietf.org/rfc/rfc3550.txt?number=3550.

[8] M. Arango *et. al.*, “Media Gateway Control Protocol (MGCP) Version 1.0,” RFC 2705, October 1999.

[9] F. Cuervo *et. al.*, “Megaco Protocol Version 1.0,” RFC 3015, November 2000.

[10] ITU-T, “Call Signaling protocols and media stream packetization for packet-based multimedia communication systems,” Recommendation H.225.0, February 1988.

[11] ITU-T, “Control protocol for multimedia communication,” Recommendation H.245, September 1988.

[12] Yu-Sung Wu, Bingrui Foo, Yongguo Mei, and Saurabh Bagchi, “Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS,” To appear in Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC ’03), December 8 - 12, 2003, Available at: http://www.ece.purdue.edu/~sbagchi/Research/Papers/cids_acsac03.pdf.

[13] M. Roesch, “Snort – Lightweight Intrusion Detection for Networks,” In Proceedings of USENIX LISA’99, November 1999.

[14] S.T. Eckmann, G. Vigna, and R.A. Kemmerer, “STATL: An Attack Language for State-based Intrusion Detection,” Journal of Computer Security, 10(1/2):71–104, 2002.

[15] CERT Coordination Center, Carnegie Mellon Software Engineering Institute, “Vulnerabilities, Incidents and Fixes,” Available at: http://www.cert.org/nav/index_red.html

[16] “SecurityFocus Vulnerabilities Archive,” Available at: http://www.securityfocus.org/bid

[17] “Prelude Hybrid IDS,” Available at: http://www.prelude-ids.org

[18] IBM Software, “IBM Tivoli Intrusion Manager,” Available at: http://www.ibm.com/software/tivoli/products/intrusion-mgr/