Detecting and Mitigating Denial-of-Service Attacks on Voice over IP Networks

Zahid Anwar, Shaya Potter, Chandra Narayanaswami, William Yurcik , Carl Gunter, Roy H Campbell
anwar@uiuc.edu, spotter@cs.columbia.edu, chandras@us.ibm.com, {byurcik, cgunter, rhc}@uiuc.edu
IBM Watson Research, Department of Computer Science
Columbia University, University of Illinois at Urbana-Champaign

Abstract— **Voice over IP (VoIP) is more susceptible to Denial of Service attacks than traditional data traffic, due to the former's low tolerance to delay and jitter. We describe the design of our VoIP Vulnerability Assessment Tool (VVAT) with which we demonstrate vulnerabilities to DoS attacks inherent in many of the popular VoIP applications available today. In our threat model we assume an adversary who is not a network administrator, nor has direct control of the channel and key VoIP elements. His aim is to degrade his victim's QoS without giving away his presence by making his attack look like a normal network degradation. Even black-boxed, applications like Skype that use proprietary protocols show poor performance under specially crafted DoS attacks to its media stream. Finally we show how securing Skype relays not only preserves many of its useful features such as seamless traversal of firewalls but also protects its users from DoS attacks such as recording of conversations and disruption of voice quality. We also present our experiences using virtualization to protect VoIP applications from 'insider attacks'.**

Our contribution is two fold we: 1) Outline a threat model for VoIP, incorporating our attack models in an open-source network simulator/emulator allowing VoIP vendors to check their software for vulnerabilities in a controlled environment before releasing it. 2) We present two promising approaches for protecting the confidentiality, availability and authentication of VoIP Services.

I. Introduction

IP Telephony is gaining rapid momentum to become a core part of our future communication infrastructure. However, before it can be widely deployed to fully replace traditional PBX or telephone networks within enterprises, many challenges need to be addressed. Besides maintaining continuous high availability and error-free operation, the IP Telephony system should be secure enough to maintain the integrity of the data streams and protect the privacy of its users.

Quality of service (QoS) and security are the two major issues that need to be addressed when voice traffic is transported over the Internet. While some traffic streams require guaranteed delivery, real-time traffic (voice, video) requires low end-to-end transmission delay across the network. Security is another important aspect that needs to be considered when using the Internet for communication. Due to the usage of shared media, user data across the Internet is susceptible to many attacks. Similar to data traffic, voice traffic is also exposed to the security attacks.

This work was funded by NSF CNS 03-05537.

Only recently [23] a hacker breached into a Newark-based company, which transmits VoIP services for other telecom businesses, using their networks to route calls for his own customers. The victim company was billed for more than 500,000 unauthorized telephone calls routed through its calling network. Eavesdropping software [17] was used on Vodafone's voice service in Greece recording conversations of the Prime Minister, security officials and other important dignitaries around the time of the Athens Olympics.

It is a common myth that it is just Voice applications based on IETF and open standards that are vulnerable and that proprietary protocols such as Skype that obscure their traffic are secure. A Chinese firm [13] recently showed how scary it can be to trust another Skype user when they reverse-engineered Skype to make free calls.

There are relatively few tools available today that check for VoIP media vulnerabilities against realistic attacker models. Vendors can claim that they encrypt every piece of information of your call but how does that effect the quality? The reason its hard to develop tools is because typically a VoIP application can not be tested alone. Fundamentally a VoIP application is at least a two-party software requiring multiple machines to work properly and there is generally a requirement of some sort of server support. For instance a SIP Voice call setup will typically require two user agents (UA), one or more SIP Proxies and optional Registration and/or Directory Servers. Similarly a H.248 or MGCP call will require Signaling, Media Gateways and Controllers apart from the actual end hosts.

In this paper we introduce VoIP Vulnerability Assessment Tool (VVAT) and describe how it was designed by extending a popular Network Simulator (J-Sim) to incorporate DoS attacker models. Using J-Sim network emulation feature to bridge between simulated and real applications we are able to test third-party VoIP applications for security vulnerabilities. For instance a SIP UA can be easily tested by peering it with a simulated UA and Proxy running inside J-Sim and mounting attacks on the transmitted traffic to check its behavior.

The remainder of this paper is organized as follows: Section II puts this work in context by providing background on existing VoIP network security tools and how they fall short. Section III describes the architecture of VVAT and its key components. Section IV describes the various attack models in detail and investigates their devastating effect on different VoIP Applications with a detailed case-study of the Skype P2P network in particular. In Section V and VI we outline how to use Session Border Controllers as a more secure alternative to Skype's concept of relay nodes and how to secure USB-Key VoIP phones that have been becoming increasingly popular recently. We end with conclusions and ideas for extending this work in Section VII.

II. Background

In this section we give an overview of existing work on tools available for testing VoIP security. Most of the work in VoIP DoS has been geared towards testing of the SIP signaling protocol, and it falls short when considering the security holes of the underlaying media protocol.

SiVus the SIP Vulnerability Scanner [16] is a VoIP vulnerability scanner developed for Miscrosoft Windows by the group at vopsecurity.org. It comprises three parts, a SIP message generator, a SIP discovery component and a SIP vulnerability scanner. It checks the SIP parser of the test application to see if it works smoothly with different variations of the SIP signalling messages. Similarly, Security Testing of Protocol Implementations (PROTOS) [14] implements a basic set of functional tests for the SIP Signaling Protocol. It checks for implementation errors and software robustness using black box functionality testing by injecting anomalies and unexpected input elements.

The SIP Torture tests [22] are a comprehensive set of SIP parser tests to check grammer that is legal in the protocol specification but likely not implemented. For instance they check whether the parser can withstand messages with whitespaces (around colons, around semicolons), empty values in unstructured headers (e.g. Subject), multiple requests in a UDP packet, extra bytes at end of UDP packet etc.

The CERT website [15] provides a somewhat comprehensive list of VoIP Vendor Vulnerabilities although not all of them are well documented. For instance there have been security bug reports listed for the Columbia SIP User Agent (sipc), Cisco Call Manager, Asterisk PBX , IPTel's SIP Express Router and Nortel's Communication Server 2000.

The Ethereal Network Analyzer [8] has some of the features we use in VVAT. For instance it can capture packets and store them to a file for later viewing and also has a easy-to-use interface to view individual headers and IP options. However unlike VVAT it does not allow real time manipulation of the packets and does not support packet injection.

Many researchers have suggested using either layer-3 or layer-4 encryption such as complex encryption mechanisms like IPSec to protect the voice traffic. Unfortunately although security mechanisms such IPSec are well know and established e.g. VPNs; they suffer from many problems (mentioned below) that render then infeasible to use with VoIP.

A. Call Setup Delay

It is impossible to use a single IPSec tunnel between the end points, as the proxies must have access to the SIP packets to set up the call. For example, if a call is being set up between two telephones, each connected to the Internet via a SIP proxy, and the call setup is to be protected end-to-end, it will be necessary to negotiate and establish three separate IPSec-protected paths for signaling: One each between the telephone and its respective proxy, and one between the proxies. According to our experiments, in the case where all signaling is protected (hop-by-hop IPSec), the call-setup delay is 24 seconds on a 100MB Ethernet which is probably unacceptable to most users.

B. Packet Expansion

A typical 40-B packet will grow up to 96-B when it undergoes tunnel-mode ESP expansion causing the ciphertext I/O bitrate to be 2.4 times less than that of the cleartext bit-rate which is not very suitable for media transformation. Besides additional overhead, IPSec and its key exchange protocol (IKE) suffer from other problems detrimental to VoIP e.g., lack of support for header compression, lack of suitable transforms and too many round trips required to complete key exchange.

C. No room for selective Encryption

IPSec security associations identify (i.e., trust) devices rather than sessions and applications. Since IPSec is a network layer solution it will encrypt everything which is a disadvantage when several sessions and applications are running on the same device, (some trusted and some not).

In previous work [3] we presented security design patterns for VoIP and in this paper we extend that by developing a tool that VoIP developers can use to check whether their products conform-to or need to incorporate those patterns.

III. Architecture

Our VoIP Vulnerability Assessment Tools (VVAT) high level architecture is show in Figure 1.

A. Packet Capture and Injection

VVAT is built on top of Libpcap [20] and Libnet [9] network tools. The packet capture library has access to all the packets in the network driver's buffer through the adapter independent dev interface. In addition it can sniff packets in promiscuous mode enabling all packets received at the NIC, regardless of whether they originated at the machine to be observed. The packets, along with the entire Ethernet, IP and UDP/TCP headers is passed up to DoS layer which examines the packets in real time and replays them (after modification if desired) and logs them to a file for off-line examination.

The DoS layer can by programmed using a rule based language, i.e, what action(s) to perform based on certain criterion. For instance it can randomly or deterministically drop, duplicate, garble or delay packets based on the header information.

```
**if** $i++ \%2$ **then**
    $ANY(udp, ip.dst$ == 10.0.12.127 AND $ip.port$ == $CURR\_SKYPE\_PORT, rand \leftarrow 50)$ $delay(20)$;
**end if**
```

The above is a very simple example of how Mallory will use VVAT to randomly delay every other packet destined to Bob 50% of the time.

We implemented the DoS engine as a Real Time Application Interface (RTAI 3.1) kernel module in Linux kernel 2.6.17, which guarantees delay precision of $100\mu sec$ over any specified packets of any flow despite the workload of the Linux kernel. To facilitate the management of the DoS Engine from user space, we also extended the netfilter/iptable mechanism in the Linux kernel. This allows the adversary to utilize the traffic flow recognition capability of the Classifier and Watermarking modules and replay a group of packets simultaneously.

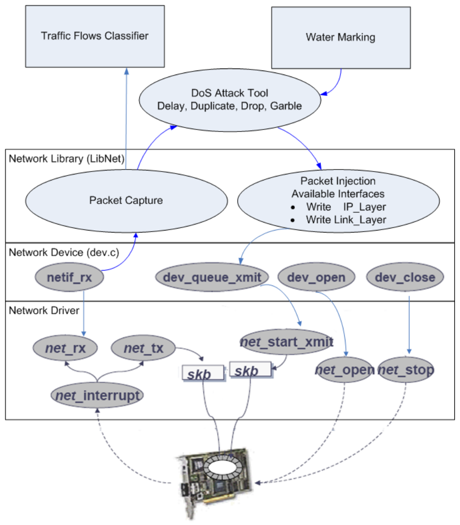

Fig. 1. VVAT High Level Architecture

B. J-Sim VoIP Components

We extended J-Sim [1] an open-source, component-based compositional network simulation and emulation environment to include VoIP RTP, SIP and H.248 stacks as components. J-Sim's network emulator allows a trade off between a real testbed and pure simulation. Typically an experiment involves network traffic that is generated by real systems and then injected into the simulator to communicate with simulated network elements. Allowing real traffic to interact with the simulator, we avoid traffic monitoring problem, and gain the ability to test real software without having to reimplement it inside the simulator. It is also easier to run batch experiments without having to configure complex VoIP servers. For more details about J-Sim emulation see [19].

C. Classification

For practical monitoring, VVAT automatically classifies the VoIP flows from the rest of the Internet traffic. It can retrieve the RTP port numbers from the SDP information provided in the signaling. If that avenue fails then it separates the packets out based on the knowledge that RTP packets have a small average packet length, packet runs or bursts (number of back-to-back packets inbound versus outbound) and relatively periodic inter-packet times typically between 30 and $40ms$.

Tested Application	Confidentiality (Codec Used)	Availability (Resulting Voice Quality)
Cisco Call-Manager	Conversation Recorded (ALaw), Identified parties	Broken
Vonage ATA	1-way Conversation Recorded (G.711, G.726 and G.729)	Broken
Skype	-	Noticably broken
LinPhone	Conversation Recorded (GSM)	Broken
X-Lite	-	Stutter
SJPhone	Conversation Recorded (iLBC)	Broken
Gizmo	-	Broken
Asterisk	Conversation Recorded (G.729)	Broken
Microsoft Communicator	-	CPU usage up 3x at 250 pps. 10x at 25000 pps. Voice Stutter
Minisip registered to openser	-	Voice Stutter. [a]
Commercial Videoconferencing Product [b]	-	Crashed (Bug reported and fixed)

[a]Enabling the SRTP (media payload encryption) option causes the application to crash because it goes over the maximum MSDU limit

[b]Requested to remain anonymous

TABLE I
VVAT DoS ATTACKS ON VoIP PRODUCTS

D. Covert Channels and Watermarking in VoIP flows

VVAT has the ability to mark flows of interest, for instance Bob talks to multiple parties simultaneously however Mallory is only interested in disrupting his conversation with Alice. Using VVAT, Mallory's accomplice can mark Alice's traffic flow with a unique watermark before it enters the Internet such that Mallory (who is on Bob's subnet) can safely identify it as it reaches Bob. There are two ways VVAT accomplishes this: (1) shifting the Interpacket Delay (IPD) distribution and (2) embedding custom packets into the stream during silence periods.

IV. VVAT DoS Attacks Arsenal

A. VoIP Threat Model

Let us examine the VoIP threat model under the classical security principles (Confidentiality, Integrity, Availability, Authentication) or CIAA. The adversary is interested in eavesdropping on conversations (breach of confidentiality property), degrade the QoS (and hence availability) of the service and finally listen-in on conversations by posing as her victim. We ignore breach of Integrity for the case of VoIP as that is automatically verifiable when listening to the other party's voice.

For now we assume that the adversary does not have any superuser and administrative privileges on any of the key network entities such as servers and switches, and so she is unable to place herself in the middle of the stream (Man-in-the-Middle) at will. In addition she wants to avoid being overly conspicuous because if her presence is detected the parties in the call may change servers or simply use other means of communication.

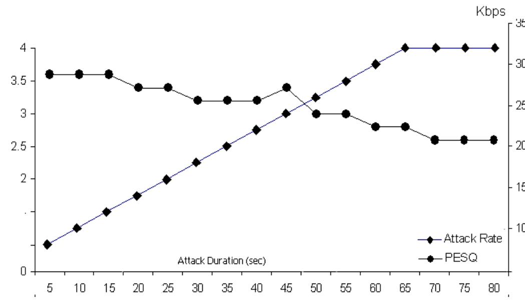

Fig. 2. Brute Force Attack on Sun's JMF Studio VoIP Application

B. Overwhelming Victim's Resources

The most basic DoS attack appears to work sufficiently well on naive VoIP applications or ones that are running on low-power and small devices. The idea is that the Adversay bombards the victim with garbage RTP packets with spoofed IP headers, overwhelming the victim's RTP buffer and corrupting the codecs that are thinking that they are processing real audio packets. Figure 2 shows the voice quality of JMF Studio degrade as the bit rate of the attack packets increases to half the bitrate of the original RTP stream.

C. Using Signaling Vulnerabilities to DoS the Media

In the case of master-slave based VoIP protocols, the server know as the Media Gateway Controller (MGC) contains all the call-setup state for the slave or the Media Gateway (MG) and commands it to perform certain tasks based on events that it sends it. For instance when a user picks up the phone the MG sends a OFF-HOOK event to the MGC. The MGC then instructs the dumb MG to play a tone for the benefit of the user. H.248 and MGCP specifications mandate the use of a Authentication Header (AH) to authenticate the MG to the MGC and vice versa.

Here is how we launched a DoS attack on the Avaz Media Gateway. The adversary Mallory has no control over the MGC-MG traffic and no way of decrypting it since she doesn't have the key. However using knowledge of the protocol itself, she can identify the *Modify Media SendOnly* message as the second-last message before the *Modify Media SendandReceive* and the subsequent periodic media packets start on the wire. By capturing the packets in one session and then later replaying it in another she can cut-off important pieces of Alice's and Bob's conversation. Figure 3 illustrates this attack. H.248 commands have unique transaction-ids that warn the MG against obeying commands with older IDs. The Avaz Media Gateway Control Protocol very predicatably uses the same sequence of transaction IDs in each session and so it is possible for Mallory to reuse commands from across multiple sessions to DoS the media, provided she is careful to use one with a transaction ID that is larger than the one the MG-MGC are currently using.

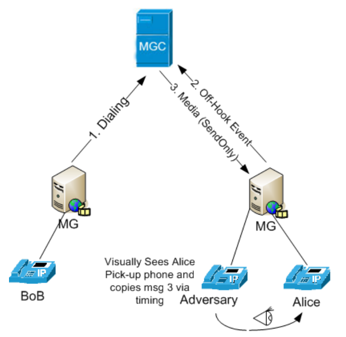

Fig. 3. Replaying Signaling to DoS the Media

If Alice and Mallory work in the same office then Mallory can also use timing to identify Signaling Messages. For instance, as soon as Alice hangs up the phone the MG sends a OFF-HOOK message to the MGC, to which the MGC sends a *Subtract* command to Bob. Mallory can use this command to cause subsequent calls to disconnect at anytime.

D. Exploiting Relayed Traffic to Launch DoS

Work on anonymity using Skype by Wang et al [25], [26] has shown that it is relatively easy to discreetly watermark Skype traffic flows such that even if the flows are made to pass through an anonymizing network (e.g. Tor) it is possible to identify the sender and receiver of the call. Research by Suh et al [27] proposes several metrics to successfully characterize and detect Skype relayed traffic. By running a Skype continuously on an end-host with a good connection to the Internet for a prolonged period of time any attacker can become a Skype relay node. Once the attacker controls enough relays he can determine the parties he is relaying the traffic for and easily mount a DoS attack.

Skype, was our most intestering victim; it being the most popular VoIP clients today with over 50 million downloads. The company uses a proprietary protocol and security mechanisms not compliant with VoIP standards to obscure the protocol.

We set up a scenario (see Figure 4) with two Skype parties in a call. The parties were on different networks and 8 hops apart. The calling party, Alice played a audio clip with a distinctive waveform (1 sec period) to Bob. Alice's and Bob's line-out were connected to a mixer that combined the two inputs into the same audio stream with different channels and displayed the channels side-by-side.

We used the ARP poison attack to enable Mallory to act as a Skype Relay. Assuming that Bob's IP is *192.168.0.12*, and the switch is *192.168.0.1*. Mallory begins by sending a malicious ARP "reply" (for which there was no previous request) to the switch, associating her computer's MAC address with *192.168.0.12*. Now the switch thinks that Mallory's computer is Bob's. Next, Mallory sends a malicious ARP reply to Bob, associating her own MAC Address with *192.168.0.1*. Now Bob's machine thinks Mallory's computer is the switch. IP forwarding is used to forward the network traffic from Bob's machine to the switch.

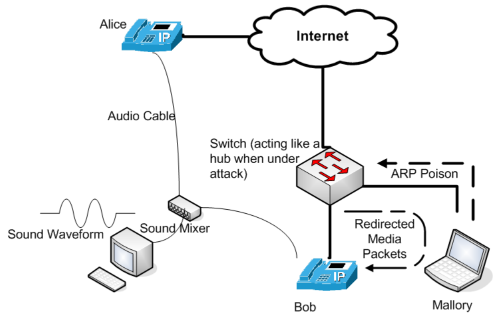

Fig. 4. Skype DoS Attack Scenario

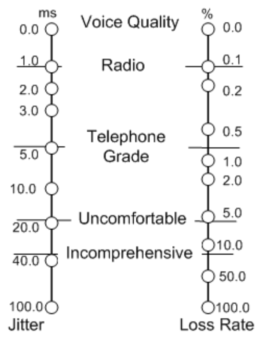

Fig. 6. The Playing of jitter and loss rate in Quality

Note that Mallory does not have full control over Skype's packets because Skype is engineered to change relays, ports and transport mechanisms if its traffic is hindered. Mallory can only launch a limited DoS attack as long as she can convince Skype that the interference she creates is due to temporary network degradations.

Figure 5 generated using Audacity 1.2.4 [4] shows the effect on the voice quality of Alice's incoming audio signal when Mallory applies VVAT's various DoS attacks on the packets she forwards. Part a) shows the original periodic signal sent by Alice. We verified that Mallory passive presence on the wire did not have any effect on the end-to-end delay and the mouth-to-ear latency (48 ms measured using the adelay tool [2]).

1) Reordering Attack: This test was designed to check if the VoIP application is vulnerable to a packet reordering attack. Given any packet flow $P_1, ..., P_n$ with time stamps $t_1, ..., t_n$ respectively $(t_i < t_j$ for $1 \leq i < j \leq n)$, the attacker swaps the set of packets $P_i, P_{i+1}, ..., P_{i+k}$ with $P_j, P_{j+1}, ..., P_{j+k}$ where $d = j - i$ and $(0 < d << n)$ and k is a small constant. Part b of Figure 5 shows the effect is more noticeable for the first part of the trace where the value of d is small. We see that a portion of the later part of the track is played early and the perceptual quality drops drastically. This means that Skype plays the packets in the order in which they are delivered. We find this behavior very unique to Skype as most other VoIP applications tend to reorder packets before delivery. In the later part of the trace a larger value of d causes Skype to apply loss concealment but still subsequently play them on arrival causing the whole waveform to be shifted all together which improves the perceptual quality somewhat. k was kept at a constant of 3 throughout the experiment. Humans tend to comprehend delayed speech more easily than speech spoken backwards.

The next two attacks focus on jitter and packet delay which play a large factor in determining service quality (see Figure 6)

2) Packet Jitter Attack: This attack was designed to test the performance of a VoIP application's Jitter Control under transient jitter and short-term delay variation. A jitter buffer increases the chances (at the expense of extra delay) that, even if some subsequent packets arrive late, enough packets will be available in the buffer for the sound to continue uninterrupted. If a packet is late and the delay is longer than the previous packet can be held, it may be dropped. If too many are dropped in a row, the speech sounds choppy. The jitter buffer can be dynamically sized to balance drops vs. delay. Only the receive buffer knows if a late packet was discarded, and can indicate this in RTCP packets to the talker. Both fixed and adaptive jitter buffers are capable of automatically adjusting to changes in delay.

Jitter buffers have not been designed with active adversaries in mind. Most root causes of jitter include CPU scheduling delays, LAN and access link congestion, router load sharing, routing table updates and timing drifts etc. The jitter buffer is like a time window with the early side representing the recent minimum delay and the late side representing the maximum permissible delay before a packet would be discarded. By tuning these min-max values the attacker can cause the window to resize frequently and drop packets.

Jitter can be measured simply as a mean of the back to back packet delay variation meaning $avg(abs(t_i \quad t_{i-1}))$. However this value corresponds to the peak to peak jitter level only if the packets arrive alternately early and late but does not work that well for other sequences such as a row of early arrivals followed by late arrivals. For this purpose we use a metric that gives us a moving average of the absolute packet delay variation, which is a more meaningful measure of the actual buffer behavior over short term period.

$$meandelay\ d_i \leftarrow \frac{(G-1).d_{i-1} + t_{i-1}}{G}$$

$$P_i = t_i - d_i \quad if\ t_i > d_i$$

$$N_i = d_i - t_i \quad if\ t_i < d_i$$

$$adjusted\ deviation \leftarrow mean(P_i) + mean(N_i)$$

Where G is a gain parameter with a suggested value [11] of 16 to give a good noise reduction ratio while maintaining a reasonable rate of convergence.

Trace c) shows the running average delay variation plotted against packet sequence number of VVATs jitter attack and its effect on Skype's audio quality and packet drops. VVAT detects when Skype relays voice packets versus silence packets to induce jitter (typically Skype's voice packets are larger $\approx 100bytes$ with shorter inter-packet spacing $\approx 60ms$ as apposed to $\approx 25bytes$ and $\approx 100ms$ for silence periods. Skype deals with low jitter deviation elegantly, smoothly playing consecutive series of delayed packets delayed by the same amount. It attempts to change port numbers and setup a new voice stream on the more reliable TCP protocol if

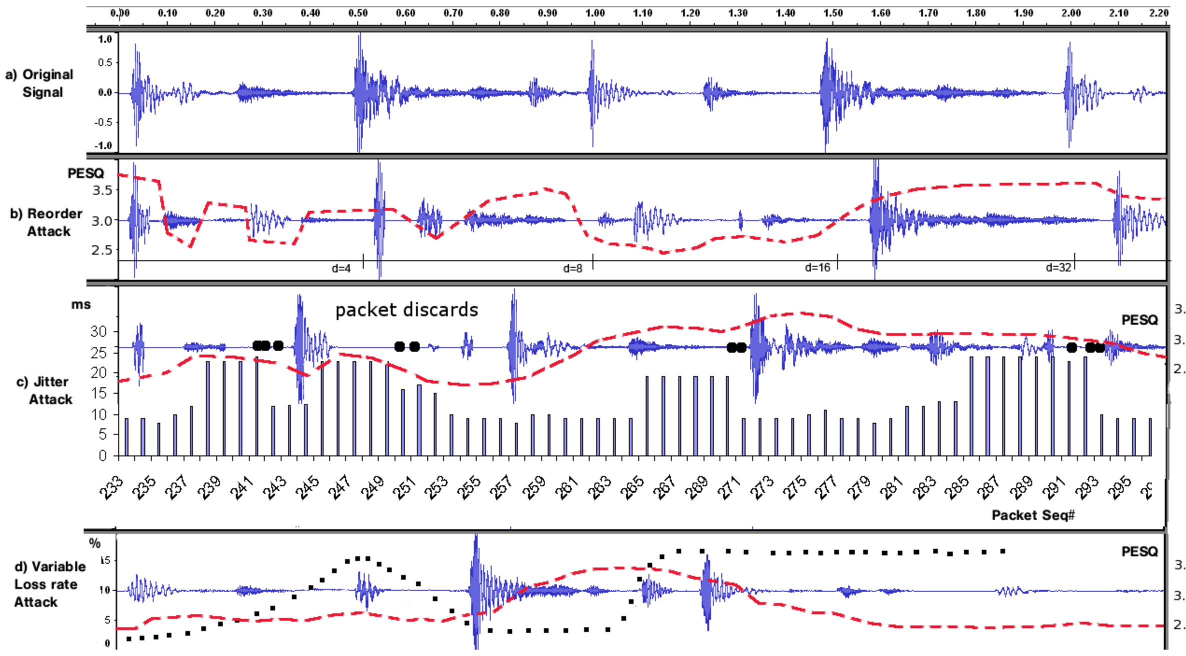

Fig. 5. MitM Attacks on Skype

VVAT delays too many packets for $> 5sec$. There are a few observations to be made here: At seq number 243 a number of packers arrive simultaneously, this being because VVAT causes the first packets to be delayed significantly and the later packets to arrive early. We see that Skype instead of dropping the first packets because they were late, instead tries to queue all packets and drops the later ones because the jitter buffer is full. At seq number 269 we see that Skype anticipating large jitter has increased the buffer size leading to less choppy sound at the expense of a longer delay. Different VoIP applications deal with this situation differently, for instance when Yahoo Voice Messenger discovers that it has more packets queuing up than it has space for, it typically plays the first packets at a faster rate giving the user a fast-forward experience.

3) Changing Loss Rate Attack: In VoIP traffic there is no time for retransmissions if packets get lost, however intelligent algorithms can be employed by the receiver to send feedback to the sender if it notices a large number of packets getting dropped. The sender can then add redundancy to the packets by piggybacking part of the already transmitted speech on the subsequent packets so that in case of loss the receiver can at least partially recover the missed speech. Trace d shows Bob's Skype adapting to dynamically changing loss rates induced by Mallory. Skype embeds feedback information inside the actual reverse path media packets. It is clear that it is able to gracefully recover with a small cost to mouth-to-ear delay from loss rates up to 25% as long as the variation is small. However with rapidly changing loss rates Skype will perform much more poorly.

It is worth noting that Mallory can cripple the feedback mechanism if it were sent separate from the media. This is achieved by blocking special RTCP packets used for reporting back statistics to the sender. Skype hides feedback packets inside the reverse path audio packets so this attack is not possible.

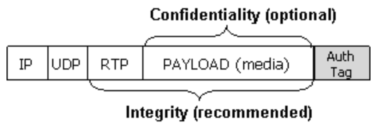

Fig. 7. SRTP Encapsulation of RTP Packets

4) Replay Attack for Encrypted Media: Some VoIP application vendors incorporate SRTP packet encryption to increase confidentiality. Authenticating each packet prior to processing discourages DoS attacks. However it is still possible to mount a successful DoS attack using replays. Note: An attacker need not decrypt packets in order to mount a successful attack. All that is needed is a mechanism to overwhelm the victims resources by having it do useless tasks preventing it the opportunity to process genuine media packets in a timely manner. Figure 7 shows the placement of SRTP confidentiality and integrity services in RTP and RTCP packets. A confidentiality service is obtained by encrypting the payload so that only the sender and receiver that are in possession of the keys can read it. An integrity service is obtained by running a one-way function on the message using a cryptographic key so that the receiver can ensure that the sender of the message possessed a secret key and that no party lacking that cryptographic key modified the message while in transit. The keys for these services are associated with the stream triple and are called SRTP cryptographic context. SRTP uses the RTP sequence number without changing the RTP header, adding it to a 32-bit SRTP rollover counter (ROC) to get the 48-bit sequence number, which is the SRTP packet index for the particular packet. The packet index is encrypted with other parameters to generate keystream segments, which is included in SRTP integrity protection as shown in Figure 7. SRTP packet-index determines the index of an invalid packet as well as a valid packet. There can be no integrity check until the authentication key is determined.

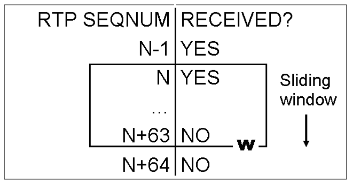

Fig. 8. Packet Replay Window

Figure 8 shows a fixed-size window on the RTP sequence number space ($SEQNUM$) that indicates if a packet with the particular sequence number has been received. Packets within the window are accepted, and a packet higher than the window ($SEQ = w' > w$) causes it to be advanced. The lower edge of the sliding window is advanced to the highest sequence number that has been received, only if the packet is successfully authenticated ($w = w'$).

The SRTP default window size is 64, which means that an authenticated packet with a sequence number that is less than 64 packets behind the highest-numbered packet or ahead of the window are discarded. Those within the window are discarded if the $RECEIVED?$ bit is set, which indicates that the particular packet has already been received.

Thus, an attacker can send a bogus packet that is within 64 packets ahead of the highest sequence number received causing it to pass replay protection. This will cause the window to be tentatively advanced; it will then fail the message authentication and the window will be restored. In that sense, the attack succeeded in forcing the receiver to run an HMAC-SHA1 hash against the packet before discarding it and restoring the replay window, SL, and SEQ to their original values. Table 1 shows the effect of the Replay Attack on three VoIP applications employing sRTP; X-Lite stuttered, while Minisip and a Commercial VoIP Phone crashed when bombarded with replayed packets within their received windows.

V. Securing the Relays

One of the solutions to securing VoIP against DoS is to secure the relays. Skype relays offer benefits such as seamless traversal of firewalls and NATs. Securing these is not plausible in the case of Skype, because in a P2P network the relays are untrusted. In fact, the very weakness of Skype security is that Skype trusts any peer that speaks the Skype language. So how do we use Skype's concept of super nodes acting as relays and at the same time establish a trust relationship between them? The answer lies with an entity in today's VoIP infrastructure known as the Session Border Controller.

A. The Session Border Controller (SBC)

In the past, enterprises have used VoIP internally with a VoIP to PSTN Gateway to talk across enterprises. Today more and more enterprises are adopting the concept of hosted VoIP services or VoIP peering (see Figure 9) where enterprises don't bother to deploy their own VoIP Servers but offload them to commercial VoIP service providers.

This has given rise to the concept of the SBCs; these (so far unstandarised) servers are located at the edge of the enterprise network and provide different sorts of services for instance:

- NAT/Firewall Traversal and Emergency 911 services
- Monitor for QoS and adherence to SLAs
- Conceal valuable route information from competitors

SBCs are put into the signaling and media path between calling and called party. SBCs intercept SIP packets and change them so that both media and SIP signaling can traverse a NAT/firewall interface. In truth an SBC is the exact equivalent of the Skype relay and is in a natural point in the VoIP infrastructure to perform security operations for its users.

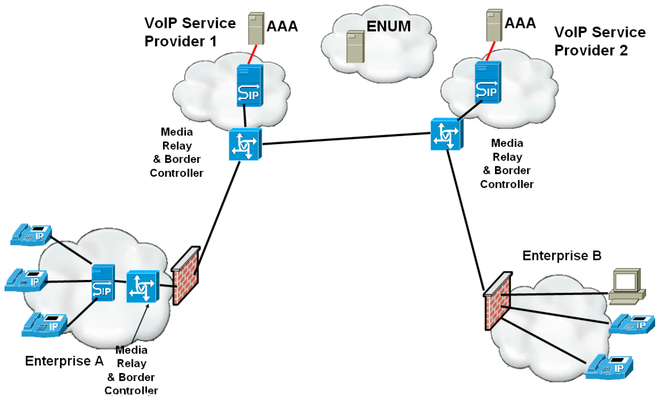
VoIP Service Provider 1
AAA
ENUM
AAA
VoIP Service Provider 2
SIP
SIP
Media Relay & Border Controller
Media Relay & Border Controller
Enterprise B
IP
SIP
Enterprise A
Media Relay & Border Controller

Fig. 9. Hosted VoIP Services. For instance Enterprise B does not have its own SIP Proxy

B. Securing VoIP at the SBC

SBCs today are "pre-standard" media relays but ultimately they will implement the "Traversal using Relay NAT" (TURN) protocol [12] and provide the capability to traverse firewalls seamlessly without having to explicitly open holes. We describe an approach for key exchange which could include the SBC in the "Circle of Trust". Let us consider the situation where the Signaling Protocol used is SIP, it will work similarly with other protocol such as H.248, H.323, IAX etc. Initially the User Agents authenticate each other using a Deffie Hellman (DH) Key Exchange between the end points. This could be done using either S/MIME (requires PKI), SDP (assuming the Signaling is secure) or inside the media channel itself using techniques such as ZRTP or EKT. After the integrity of DH key setup is verified by ABCs and XYZs digital signatures, the endpoints connect to the SBC TURN relay using TLS and authenticate themselves. As a result the endpoints receive a port and IP address from the SBC to relay the media. An extra message could easily be added to TURN to communicate the session key to the relay. Once the relay knows the key it can authenticate traffic on behalf of the client. This is useful because normally the SBC will be running on a dedicated machine with more resources than the end hosts. End hosts such as smart phones and PDAs will typically run out of resources much faster under a DoS attack than an

SBC managing multiple connections. In addition the SBC can employ other security measures that are normally difficult at the application layer for instance: Media Rate Limiting. The SBC can read the codec information to determine the rate of the media channel and drop packets if the packet rate exceeds the allocated rate. In the case where a Adversary injects DoS packets into a connection, this will prevent congestion at the SBC and protect the infrastructure. SBCs could also help the application check for replay attacks by keeping a local replay list and throwing away duplicates in the range of the current RTP receive window.

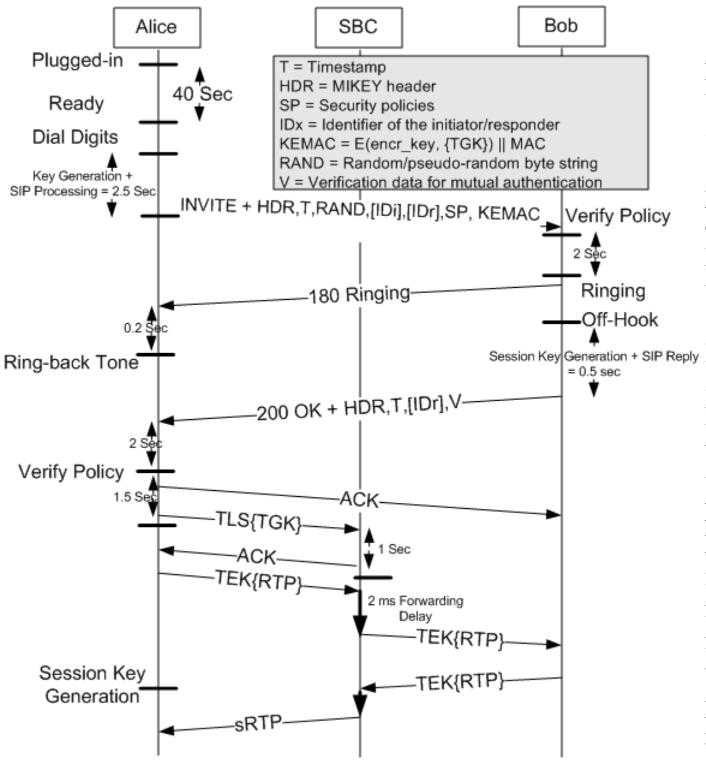

Fig. 10. Sequence Diagram of VoIPPod and SBC setting up a VoIP Session

VI. Handling Insider Attacks: Bugs in the Telephone Set

In this section we address the problem of Authentication in the CIAA threat model. Hackers can be insiders, with easy access to data files, conference phones and computers. The modern day version of wiretapping and planting bugs in the phone is installing Spyware, Trojans, and voice loggers in the OS while the victim is away. Recall the XCP program [7], developed by First4Internet in Britain and used on music CDs by Sony BMG to restrict copying and sharing, that acts like virus software and hides deep inside a computer where it leaves the backdoor open for other viruses.

A. VoIP Pod Architecture

USB VoIP phones are becoming very popular [24], increasing the risk of having your phone, address book, passwords etc stolen. We extended the idea of SoulPad (Reincarnating PCs with Portable SoulPads) [5] for VoIP appliances and present our experiences with developing a secure and portable virtualized VoIP Phone on a WiFi memory stick. We divide a 1 GB USB Key into four main partitions: (Figure 11) a small typically ($< 100MB$) Boot Partition, an encrypted swap partition, a Linux ext3 partition and a hidden and encrypted data partition. We modified the boot-up sequence scripts to load USB drivers prior to mounting the file system and to boot up directly from the USB drive. Partition 3 contains a very small stripped down version of the Autoconfiguring KNOPPIX Distribution ($250MB$) responsible for starting up Xen VMM and the encrypted VoIP Phone image stored on partition 4. Network Setup scripts probe to check if the USB is plugged to a machine with an active Ethernet connection otherwise it uses XSupplicant [21] WiFi Libraries to associate the Wireless NIC on the stick to the AP selected by the user. The entire Linux Partition is mounted as a Read Only UNIONFS [18] file system preventing download and permanent storage of malicious code. To protect the VoIP Phone if a VoIPPod is misplaced or stolen, we encrypt the disk partition that holds the VM image using the AES128 block cipher. We used the publicly available LUKS [10] cryptosetup loop-aes package in our implementation. The advantage of using LUKS is that the partition can by easily and securely mounted in a a Microsoft Windows OS using the FreeOFTE tool [6]. The encryption key is generated by hashing a user supplied pass phrase. On bootup, the user is prompted for a pass phrase which if supplied incorrectly, results in the resulting hash not corresponding to the AES key and the mount operation to fail since the decrypted data will not correspond to a valid file system. In order to defeat brute force attacks that attempt to guess the pass phrase, the loop-aes package requires the pass phrase to be at least 20 characters long. The AES key is retained in kernel memory while the phone is running and is erased when the partition is unmounted. The swap partition is similarly encrypted to prevent viewing of swaped-out pages. The key is however auto-generated for every session since swap state is not preserved across boot cycles. VoIPPod doesn't write to internal disk and therefore, there is no risk of leaving sensitive data on the PCs persistent storage after disconnecting.

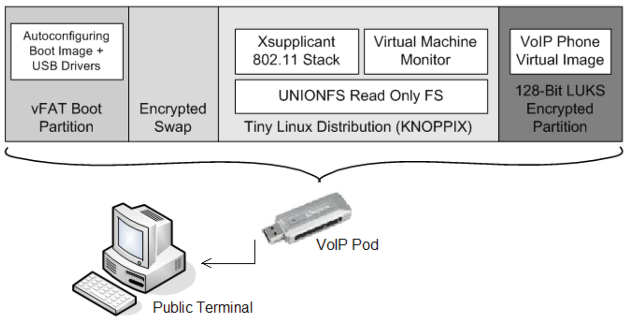

Fig. 11. A Secure and Portable VoIP Soft Phone

B. VoIP Pod Performance

Our experiments show that running off a USB 2.0 (480 Mbps) has no noticable effect on VoIP performance. Whereas

USB data transfer (as apposed to IDE) might be an issue for data intensive applications, it makes no difference to VoIP applications that are loaded and run entirely from memory. Virtualization does have its processing overheads though, which manifest itself as a fixed increased delay of 3-4 ms (on VMWare Server 1.0) adding to the mouth-to-ear latency. Fortunately there is no effect on jitter or on loss, so this extra delay is tolerable for the security it provides. Figure 10 shows the detailed message exchange and the processing delay when Alice and Bob were using VoIP Pod with Intel Pentium 4, 1 GHz laptops running Linux 2.6. Libcrypto was used for the cryptographic operations.

In addition to compartmentalizing the application via a virtual machine hyper-visor and isolating it from Operating System bugs, malware and spy ware, VoIPPod allows the convenience of carrying your VoIP phone, contacts and settings on person on a key ring.

VII. Conclusion and Future Work

In this paper we presented VVAT-a DoS attack simulation tool. VVAT makes it easy to test third party VoIP applications by pairing them with simulated VoIP components. Pitted against a number of VoIP applications, VVAT showed that many do not employ any confidentiality measures allowing the passive recording of conversations. Even applications that do employ encryption are still vulnerable to DoS Attacks whereby the Adversary can disrupt the conversation sufficiently by brute-forcing the application buffer. Skype is also vulnerable to attacks despite its use of a proprietary protocol. A VoIP vendors first line of defense is to incorporate a packet encryption tool such as sRTP and to use trusted Skype-like relays to forward voice traffic. SBCs are a natural equivalent of Skype relays that use open standards. We demonstrate a key-exchange mechanism to bring the SBC into the 'circle of trust'. VoIP is also vulnerable to 'insider' attacks where attackers can install malware on the machine and have victims' conversations streamed to them on convert channels. To this effect we developed VoIPPod - a VoIP application which boots and runs off of a USB memory key inside a VM compartmentalizing it from viruses and Trojans.

A. Future Work

We are also looking at designing an attack that exploits the sRTP Diffie-Hellman key exchange. In this attack, Mallory can intercept Alice's public value and send her own public value to Bob. When Bob transmits his public value, Mallory substitutes it with her own and sends it to Alice. Mallory and Alice thus agree on one shared key and Mallory and Bob agree on another. This vulnerability is present because Diffie-Hellman key exchange does not authenticate the participants.

Using VoIP over wireless connections with its lossy, contention based channel access raises many interesting research questions in improving performance (using larger voice packets for instance). The 802.11E extension for wireless is specially designed with VoIP in mind to give higher priority to voice packets. However having a shared medium, the ability to jam signals and multi-hop scenerios open many new avenues for the adversaries interested in mounting DoS attacks. We plan to explore this area further.

While emulation software like VVAT can be useful for unearthing poorly written VoIP programs, their limitations must also be understood. As explained by Dijkstra, "Program testing can show the presence of bugs, but never their absence." We have not yet addressed the issue of a designing a jitter buffer algorithm that recognizes the presence of an attacker (as apposed to normal congestion) and raise an alarm. This looks like a promising area to explore in the future.

References

[1] J-Sim. http://www.j-sim.org/.

[2] ADelay. Measure the delay between two audio channels. In *www1.cs.columbia.edu/ IRT/ software/ adelay/ adelay.html*.

[3] Zahid Anwar, William Yurcik, Ralph E. Johnson, Munawar Hafiz, and Roy H. Campbell. Multiple Design Patterns for Voice over IP (VoIP) Security. In *Workshop on Information Assurance (WIA) , held in conjunction with 25th IEEE International Performance Computing and Communications Conference (IPCCC)*, April 2006.

[4] Audacity. The Free, Cross-Platform Sound Editor. In *audacity.sourceforge.net/*.

[5] Ramon Caceres, Casey Carter, Chandra Narayanaswami, and M. T. Raghunath. Reincarnating PCs with Portable SoulPads. 2005.

[6] Sarah Dean. FreeOTFE: A free "on-the-fly" transparent disk encryption program for MS Windows 2000/Windows XP. 2006. www.freeotfe.org/.

[7] Charlie Demerjian. The Sony DRM Scandal". 2005. www.theinquirer.net/? article=27426.

[8] Ethereal. Ethereal Network Analyzer. In *www.ethereal.com*.

[9] Packet Factory. Network Library. In *www.packetfactory.net/libnet/*.

[10] Clemens Fruhwirth. TKS1 - An anti-forensic, two level, and iterated key setup scheme. July 2004.

[11] Network Working Group. RFC1889 Real Time Control Protocol. 1996.

[12] IETF. Traversal using Relay NAT. 2004. www.jdrosen.net/ papers/ draft-rosenberg -midcom-turn-02.html.

[13] The Skype Journal. Skype responds to "Skype protocols opening up, ready or not.". 2006. www.skypejournal.com/blog/archives/2006/07/.

[14] Rauli Kaksonen. A Functional Method for Assessing Protocol Implementation Security. In *Technical Research Centre of Finland, VTT Publications 447. 128*, volume 447, page 128.

[15] Center of Internet Security Expertise (CERT). CERT/CC VoIP Vulnerabilities. In *www.kb.cert.org/vuls/id/528719*.

[16] Voice over Packet Security Forum. SiVus, The VoIP Vulnerability Scanner. 2004. www.vopsecurity.org/html/tools.html.

[17] The Register. Greece rocked by mobile phone tapping scandal. 2006. www.theregister.co.uk 2006/ 02/06/.

[18] Josef Sipek. UnionFS: User and Community-oriented Development of a Unioning Filesystem. 2006.

[19] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang. J-Sim: a simulation and emulation environment for wireless sensor networks. *IEEE Wireless Communications Magazine*, April 2005.

[20] Sourceforge. Packet Capture Library. In *sourceforge.net/projects/libpcap/*.

[21] SourceForge. Open Source Implementation of IEEE 802.1X. 2006. open1x.sourceforge.net/.

[22] Robert Sparks. SIP Torture Tests. In *(draft-ietf-sipping-torture-tests-04)*.

[23] New York Times. Hacker Said to Resell Internet Phone Service. 2006. www.nytimes.com 2006/06/07/technology/07cnd-voice.html.

[24] Vonage. V-Phone. vonage.com/device.php? type = VPHONE.

[25] X. Wang, S. Chen, and S. Jajodia. Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet. In *ACM CCS*, 2005.

[26] X. Wang and D. Reeves. Robust Correlation of Encrypted Attack Trac Through Stepping Stones by Manipulation of Interpacket Delays. In *ACM CCS*, 2003.

[27] X. Wang and D. Reeves. Characterizing and Detecting Skype-Relayed Traffic. In *INFOCOM*, 2006.