Multiple Design Patterns for Voice over IP (VoIP) Security

Zahid Anwar[†] William Yurcik[‡] Ralph E. Johnson[†] Munawar Hafiz[†] Roy H. Campbell[†]

[†]*Department of Computer Science*
[‡]*National Center for Supercomputing Applications (NCSA)*
University of Illinois at Urbana-Champaign
{anwar,johnson,mhafiz,rhc}@cs.uiuc.edu byurcik@ncsa.uiuc.edu

***Abstract*– Design patterns capture software solutions to specific problems that have evolved over time and reflect many iterations of work. Documenting such patterns promotes proven design and software reuse. There has been a growing amount of work documenting design patterns for security, however, little work specific to VoIP security. In 2005 NIST released a report on recommendations and best practices for securing VoIP, however it lacks the structure, terminology, and ease-of-understanding needed for both technical and non-technical audiences that is an inherent feature of design patterns.**

In this paper we document three design patterns for VoIP implementations related to specific security problems: (1) secure traversal of firewalls and NATs; (2) detecting and mitigating DDoS attacks; and (3) securing against eavesdropping. With many VoIP vendors rushing products to market with overlapping functionality and requirements for interoperability, documenting design patterns is poised to become an important part of secure programming processes for VoIP.

***Index Terms*– security design patterns, VoIP security, threat modeling, secure traversal of firewalls and NATs, Internet telephony**

I. INTRODUCTION

The continuous number of high-profile Internet security breeches reported in the mass media show that despite an emphasis on security processes that there is still a gap between theory and practice. Not only is there a need to develop better software engineering processes but also theoretical security improvements need to find their way into real systems.

Software design patterns are defined as *"descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context"* [5]. As software design patterns have proven their value in the development of production software[1], they are a promising new approach to help in both the theoretical development and practical implementation of better security processes [3,12,15,16]. First, many/most software developers have only a limited knowledge of security processes and patterns are a proven way to improve their understanding. Second, patterns work against "reinventing-the-wheel" to promote learning best practices from the larger community to save time, effort, and money with easily accessible and validated examples. Third, code can be reused since the same security patterns arise in many different contexts. Fourth, *AntiPatterns* or common security failures are valuable examples of what not to do [18].

A growing number of security design patterns have already been documented including patterns for services (firewalls, mailers) and common security functions (authenticators, key management) [16]. In this paper we focus on Voice over IP (VoIP) security where security design patterns may prove exceedingly useful. Internet telephony or VoIP has grown in importance and has now passed the tipping point – in 2005 U.S. companies bought more VoIP phones than ordered new POTS lines [20]. However, with the powerful convergence of software-based VoIP to enable new functionality to store, copy, combine with other data, and distribute over the Internet also comes security problems that need to be solved in standard ways in order to ensure interoperability. This is further complicated by the fact that various vendors competing for market share currently drive VoIP security.

Given the importance of VoIP security, we are only aware of only two other efforts for VoIP security design patterns, a chapter within [15] and an unpublished M.S. thesis supervised by Eduardo Fernandez of Florida Atlantic University.

The remainder of this paper is organized as follows: Section II summarizes previous work by reviewing a recent NIST report on VoIP security. Section III briefly presents three design patterns for VoIP security. We end with a summary and conclusions in Section IV.

[1] Java APIs, OpenStep libraries, and the Microsoft Foundation Classes all use catalogued design patterns.

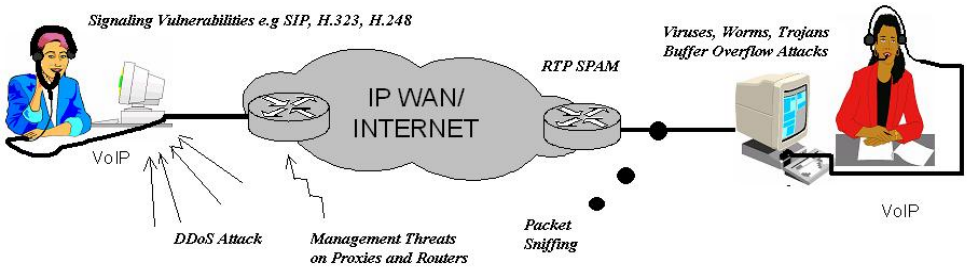

Figure 1. VoIP Infrastructure Vulnerabilities

II. Review of NIST VOIP Report

NIST released a report on VoIP security in January 2005 [10]. This report elaborates on various aspects of securing VoIP and the impact of such measures on call performance. The report argues that VoIP performance and security are not seamlessly compatible; in certain areas they are orthogonal. We briefly review this report and group VoIP infrastructure threats into three categories as depicted in Figure 1: (1) protocol, (2) implementation, and (3) management.

A. Quality of Service (QoS) Issues

The description given in the NIST report about QoS issues is primarily based on the work by Goode [6], Barbieri [1], and Chuah [4]. A VoIP call is susceptible to latency, jitter, and packet loss. ITU-T recommendation G.114 [7] has established 150 ms as the upper limit on one-way latency for domestic calls. If Goode's latency budget is considered, very little time (< 29 ms) is left for encryption/decryption of voice traffic. QoS-unaware network elements such as routers, firewalls, and Network Address Translators (NAT) all contribute to jitter (no uniform packet delays). Use of IPsec both contributes to jitter and reduces the effective bandwidth. VoIP is sensitive to packet loss with tolerable loss rates of 1-3%; however, forward error correction schemes can reduce loss rates.

B. Signaling and Media Protocol Security

SIP (Session Initiation Protocol) (RFC 3261) and H.323 [8] are the two competing protocols for VoIP signaling. H.323 is an ITU-T umbrella of protocols that supports secure RTP (SRTP) (RFC 3711) for securing media traffic, and Multimedia Internet Keying (MIKEY) (RFC 3830) for key exchange. SIP supports TLS and S/MIME for signaling message confidentiality and SRTP for media confidentiality.

C. Firewalls and NATs

RTP is assigned a dynamic port number that presents a problem for firewall port management. A firewall has to be made aware of the ports on which the media will flow. Thus a stateful and application-aware firewall is necessary. However, if a client is behind a NAT, call establishment signaling messages transmit the IP address and RTP port number that is not globally reachable. NAT traversal protocols like STUN (RFC 3489), TURN (RFC 2026), and ICE (14) are necessary to establish a globally routable address for media traffic. For protocols that send call setup messages via UDP, the intermediate signaling entity must send to the same address and port from which the request arrived.

C. Encryption and IPsec

IPsec is preferred for VoIP tunneling across the Internet, however, it is not without substantial overhead. When IPsec is used in tunnel mode, the VoIP payload to packet size ratio for a payload of 40 bytes and RTP/UDP headers drops to ~30%. The NIST solution to avoid queuing bottlenecks at routers due to encryption is to perform encryption/decryption solely at endpoints. SRTP and MIKEY are specified for encrypting media traffic and establishing session keys respectively.

D. Categorizing VoIP Threats

The threats faced by a VoIP are similar to other applications including: unwanted communication (spam), privacy violations (unlawful intercept), impersonation (masquerading), theft-of-service, and denial-of-service. Table 1 groups these threats into protocol, implementation, and management categories.

Table 1. Categorizing VoIP Threats

Protocol	
Signaling, Media Confidentiality, Integrity	end-to-end protection as well as hop-by-hop (Proxies might be malicious)
Configuration, Confidentiality, Integrity	most VoIP devices are managed remotely
Identity Assertion	Users concerned about whether they are talking to the real entity as opposed to a 'phished' entity
Reputation Management	
Implementation	
Buffer Overflow, Insecure Bootstrapping.	
Management	
Access Control	protection against unauthorized access to VoIP servers and gateways
Power Failures	

D. Is the NIST Report Complete?

In four key areas we find the NIST report incomplete. First, the NIST report cites results that the SHA1 hash algorithm throughput is less than the throughput of DES/3DES for a VoIP packet stream. This result is counter-intuitive since encryption/decryption algorithms are generally believed to require more processing than hashing algorithms.

Second, the Mean Opinion Score (MOS) is a standardized quantitative measure of human speech quality at the destination end of a voice circuit. MOS uses subjective tests that are averaged to calculate an indicator of system performance. The NIST Report does not use MOS which is a useful metric for balancing security versus performance tradeoffs.

Third, greater payload compression means that the codec employs temporal relationships between the voice blocks. It is this temporal relationship that is sensitive to packet loss. However, this is not clear from the NIST report when it says "greater payload compression rates resulted into higher sensitivity to packet loss" [10].

Fourth, the NIST report does not anticipate the use of VoIP as a SPAM DoS tool. While Email spam relies on SMTP servers for transmission, VoIP RTP packets have no such constraint. Qovia [11], a company that sells tools for VoIP monitoring and management, recently applied for two patents on technology to both broadcast *and* block messages using VoIP.

While VoIP has threats, in comparison traditional phone service has dealt with many threats over many years. For example, tapping a landline at a wirebox and eavesdropping by tuning in a cordless phone frequency is currently easier than spoofing VoIP packets. VoIP companies such as Vonage and AT&T have taken only preliminary precautions at this point (firewalls). The VoIP Security Alliance (VOIPSA) [19] has been organized to improve security awareness and form consensus on "Best Practices".

III. VOIP SECURITY DESIGN PATTERNS

While the NIST report missed some points, it does provide a general summary of VoIP security problems. However, the NIST report does not describe specific solutions used to overcome the problems identified. In this section we describe software design patterns that have emerged as implemented solutions to specific VoIP security problems. In the first pattern we describe the ad hoc but effective techniques used by most VoIP vendors to traverse firewalls and NATs.[2] In the second pattern we describe how to keep "Man-in-the-Middle" attacks from disrupting VoIP connections. In the third pattern we describe how to protect VoIP against eavesdropping.

Software design patterns typically have a presentation format that includes: (1) UML diagrams of the structure and dynamic interactions of the objects that constitute the patterns, (2) examples of the patterns in use, (3) pointers to related patterns, and (4) sample code implementing the pattern. We adapt this format due to space limitation to focus on the four essential elements of a pattern: (1) pattern name, (2) problem, (3) solution, and (4) consequences [5]. We begin each design pattern with a name and example scenario with accompanying assumptions. Given the example context, we present problem(s) matched with corresponding solution(s), and consequences – all of which distilled from the VoIP community.

A. VoIP Security Design Pattern One

Secure Traversal of Firewalls or NATs for VoIP: allows clients using private IP addresses hidden behind firewalls/NATs to be able to make VoIP calls to other clients without the need to modify intermediate firewalls or NATs or making assumptions about device types.

Example: Alice works at a large organization scheduling meetings between teams distributed at offices around the world using the telephone. Her manager decides that she should use a new VoIP phone for cost savings. Alice is not concerned about voice quality in her short conversations.

As shown in Figure 2, the organization uses firewalls with strict policies. Alice's new VoIP phone uses SIP but it is giving her problems. She is trying to call Bob at another office behind a firewall. Alice had been warned that firewalls are a common problem with VoIP phones. When the called party picks up the phone the two people often cannot hear each other. In Alice's case, Bob's phone does not even ring when she calls.

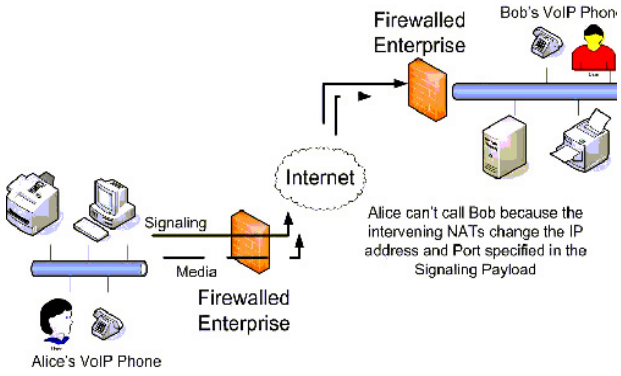

Figure 2. VoIP Between Firewalled Networks

[2] *Skype*, the world's most popular VoIP service provider with 38 million software downloads representing about 5 percent of all Internet users implements the techniques described in pattern one [17].

Alice tries to switch to phones that use other protocols (e.g. H.323) but the same problem occurs. She contacts her organizational support staff to request opening certain firewall ports but is turned down. After hours of debugging, Alice and the technical support staff of the VoIP provider deduce that the NAT device her organization uses is the most stringent type (i.e. a symmetric NAT that changes its IP/port configuration based on both caller and callee identities).

Problem: Setting up a VoIP call has two major parts. First, a signaling protocol is used to set up a call and play Dial and Ring tones. Subsequently if the called party goes off-hook then this protocol negotiates address/port and then the data protocol takes over to exchange voice until the call is torn down.

Firewalls and Network Address Translators (NATs) are located at the edge of most all enterprise networks. Often software-based firewalls and NATs are bundled in residential DSL packages as well, so this problem affects both business users and residential users. The problem starts with how to locate a client that is behind a firewall? How to determine if they are even online?

Signaling between clients contains details of the private IP addresses and ports that the clients want to use for the media flows. When the clients attempt to use these private addresses to send/receive media, the connection fails because they are not routable. Some solutions such as TURN and STUN have been proposed to help solve this problem. However, they are incomplete because they are designed to work with data only (assuming signaling is working) and do not work with every type of NAT.

The TURN protocol requires TURN capability in the actual client and a trust relationship based on shared credentials [13]. A VoIP phone or software package may include a STUN client, which will send a request to a STUN server. The server then reports back to the STUN client what the public IP address of the NAT router is and what port was opened by the NAT to allow incoming traffic back in to the network. The response allows the STUN client to determine what type of NAT is in use, as different types of NATs handle incoming UDP packets differently. This will work with a full cone NAT (address binding remain constant for all outgoing connections), but requires some special treatment with restricted cone NATs that only allow connections initiated by firewalled machines. STUN will not work with symmetric NATs (which create new bindings based on *each* source and destination pair). Unfortunately, symmetric NATs are found in many enterprise networks.

Solution: A Global Directory Index (GDI) maintains a list of all online clients. Certain clients are selected that are accessible from the public network to act as relays. When a client comes online it registers with the GDI which does not save the contact information provided in the Register message but rather saves the real address. Subsequently, the GDI and client exchange keep-alive packets with the GDI below the NAT binding expiration time threshold. When a permanent link is open between a client and the GDI, a VoIP signaling session can then be negotiated at any time.

When two clients wish to communicate, the caller tries to contact the called party directly. However, if the called party is protected by a NAT, then the called party's computer is asked by the GDI to connect in the reverse direction back to the caller's computer. If either of these connections succeeds then the call is established using the direct connection that provides the lowest-latency connection possible.

If both parties to the call are behind restrictive firewalls, then neither party will be able to reach the other directly. The GDI then chooses a third party (relay) who is reachable by both parties. In this case, both the caller's and the called party's computers establish a direct link to the relay that will forward data packets between the two parties. When calls are relayed by third parties, the entire contents of the call (including any voice conversations, text messages, or file transfers) are encrypted between the caller and the called party.

Structure: The signaling sequence diagram in Figure 3 shows the interaction between different entities in the system. During startup, each VoIP client registers with the GDI. The GDI records the client's actual public IP address and port. Subsequently it exchanges keep-alive messages with the client to ensure that the bindings remain open. If the callee happens to be behind a restricted-cone NAT (where the private IP may only participate in a connection that it initiates) then it asks the callee to call back the caller. In the case where RTP media cannot flow directly between parties the GDI chooses a public relay that tunnels the media for them. With the most global view, the GDI is in the best position to select relays for shortest path routing, low latency routing, and load balancing.

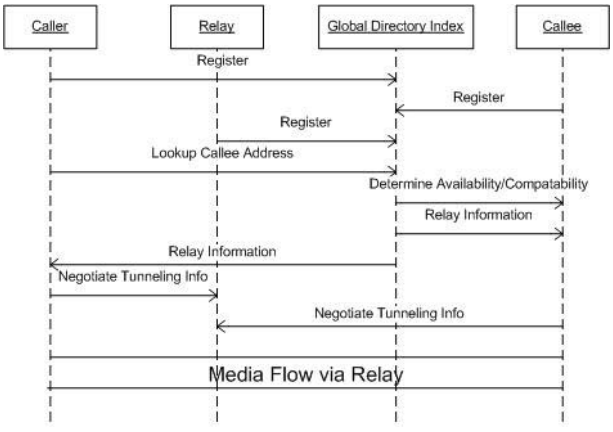

Figure 3. VoIP Signaling Sequence Diagram

Implementation: There are two parts to maintaining a VoIP connection - signaling and media. VoIP will normally open separate ports for each part. If the client is behind a restrictive firewall, the client will have to maintain bindings for the two separately.

VoIP signaling protocols can be roughly divided into two main categories, client-server and point-to-point. SIP and H.323 are two popular examples of point-to-point protocols. SIP which is relatively lightweight and flexible For example, the HTTP text-based protocol uses an *"INVITE"* message to request a session with a successful response being *"200 OK"*. These simple messages can be used to establish communication between a SIP client and a GDI. In the event that the caller cannot access the callee directly, the GDI can ask the callee to send a "*RE-INVITE*" message to the caller.

For client-server based VoIP protocols, the controlling entity is known as a "Call Agent" which manages all the signaling between the media streaming devices known as Gateways (e.g. MGCP and H.248). In such a scenario the role of the GDI can be played by the Call Agent resulting in minimal change to the protocol.

The Session Description Protocol (SDP) is used by all VoIP signaling protocols to exchange parameters particular to the session (RFC 2327). Parameters include but are not limited to: IP address, port number, frame rate, compression type, and encoding. The NAT bindings should be exchanged via the SDP protocol. The example in Figure 4 shows a SIP message encapsulating a SDP descriptor (Lines 13 to 20). The highlighted values indicate the IP address and port that need to be changed in either the public address provided by the NAT binding or the address of the relay.

```
001 INVITE sip:12125551212@211.123.66.222 SIP/2.0
002 Via: SIP/2.0/UDP 211.123.66.223:5060;branch=a71b6c57-507c77f2
003 Via: SIP/2.0/UDP 10.0.0.1:5060;received=202.123.211.25;rport=12345
004 From: <sip:2125551000@211.123.66.223>;tag=108bcd14
005 To: sip: 12125551212@211.123.66.222
006 Contact: sip: 2125551000@10.0.0.1
007 Call-ID: 4c88fd1e-62bb-4abf-b620-a75659435b76@10.3.19.6
008 CSeq: 703141 INVITE
009 Content-Length: 138
010 Content-Type: application/sdp
011 User-Agent: HearMe SoftPHONE
012
013 v=0
014 o=deltathree 0 0 IN IP4 10.0.0.1
015 s=deltathree
016 c=IN IP4 10.0.0.1
017 t=0 0
018 m=audio 8000 RTP/AVP 4
019 a=ptime:90
```

Figure 4. Session Description Protocol (SDP)

Once the signaling determines that the parties are ready to talk, a new channel is opened for media binding. Since the GDI can only be used to maintain the NAT binding for the signaling this binding has to be maintained in some other way.[3] RTP is normally used to transfer the media payload between the clients. The receiving address and port number have to be determined by the client itself using a signaling protocol such as SIP, H.323, MGCP, H.248 etc. The frequency of media packets is a typically much higher than that of the signaling and the binding is maintained automatically. This way there is no need to modify the RTP protocol except to use the same port for incoming and outgoing media. The only way the binding can be broken is if there are long silent pauses in between conversations and the media packets are suppressed to preserve bandwidth. It is desired that silence suppression not be used as a feature when using this pattern, however, silence suppression is extremely popular with VoIP vendors because conversations typically consist of ~50% silence. Disabling silence suppression will ensure the UDP bindings at the NAT are maintained.

Known Uses: *Skype* uses peer-to-peer networking with super nodes as relays to overcome the NAT traversal problem [2]. *eNat* software runs on the client device and allows the popular *MSN Messenger* to allow voice chat behind firewalls [9]. It basically acts as a proxy diverting all signaling and media through itself and through special ports it asks the user to open in the firewall. This will not work in an enterprise setting where a user does not have control of the firewall.

Consequences: This pattern has these advantages:

1) End users do not need to be aware of NAT and firewall configurations
2) Provides increased security over opening ports or tunneling through firewalls
3) The overlay nature of this pattern distributes relay load over multiple clients
4) Works for groups of users, however, the conference size is limited by relay bandwidth

This design pattern has the following disadvantages:

1) Relays increase bandwidth consumption
2) The GDI is a single point of failure
3) Added complexity

B. VoIP Security Design Pattern Two

Detecting and Mitigating DDoS Attacks Targeting VoIP: allows key components in a VoIP infrastructure to detect and mitigate Distributed-Denial-of-Service (DDoS) attacks meant to overwhelm either client and/or server resources and disrupt VoIP operations.

Example: Alice tries to call Bob on her VoIP telephone with an important message as depicted in Figure 5. An attacker anticipates Alice's call attempt

[3] media never goes through the GDI but rather directly between the parties and the relay

and sends a specially crafted messages to Alice's ISP server causing it to over allocate resources such that Alice receives a "service not available" (busy tone) message.

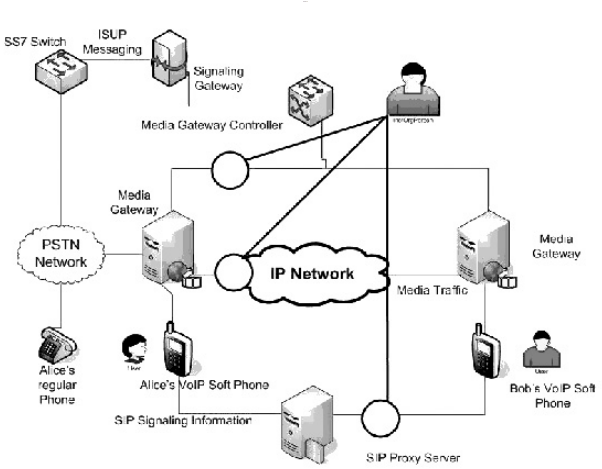

Figure 5. VoIP Environment to Consider DoS Attacks

Alice switches to her backup line that uses the SIP protocol that does not signal through a central server. Her phone makes a direct connection to Bob but before Alice can convey her message an attacker sends a special 'BYE' message to Bob's VoIP phone pretending to be Alice which prematurely ends the connection.

Alice is persistent and dials Bob again. This time the attacker intermittently sends garbage voice packets to Bob's phone in between those of Alice's voice packets. Bob's phone is so busy trying to process the increased packet flow that the jitter (delay variation) causes any conversation to be incomprehensible.

In this pattern we assume that the vendor has implemented the minimal set of recommended VoIP security requirements [10]. Megaco recommends security mechanisms in the underlying transport mechanisms such as IPSec. Implementations of the protocol using IPv4 are required to implement the interim AH scheme.

MGCP (RFC 3435) and Megaco/H.248 (RFC 3261) are control protocols designed to centrally manage Media Gateways (MG) deployed across a VoIP infrastructure. A MG executes commands sent by the centralized Media Gateway Controller (MGC) and is designed to convert data between PSTN to IP, PSTN to ATM, ATM to IP, and also IP to IP. MGCP and Megaco/H.248 can be used to set up, maintain, and terminate calls between multiple endpoints, while monitoring all of the events and connections associated with those endpoints from the MGC. The MGC is a key component in the entire infrastructure as it can control multiple MGs each with its own many endpoints (VoIP users). In addition it talks to other MGCs using SIP if a call warrants a connection between an endpoint on an MG controlled by one MGC and endpoint on an MG controlled by a different MGC.

Occasionally some endpoints have a feature which enables them to make a call directly to another endpoint without having to go through an MG or MGC using SIP. Usually this requires an address lookup from a SIP Proxy server and the rest of the signaling and media is handled by the endpoints themselves.

Problem 1: An attacker can disrupt Alice's Media Gateway. MGs are dumb clients in the MGC-MG relationship, not holding call state and carrying out instructions received from the MGC without validity checks. Although an attacker cannot pretend to be the MGC owing to the AH check required in the authentication header of the Megaco protocol, they can mount a replay attack on the MG when they detect that Alice is about to make a call.

An attacker has a number of options available after Alice is detected making a call. They can replay with any one of the Megaco commands listed in Table 2 spoofed off the MGC outbound links earlier in the communication. By replaying any of these messages to Alice during her call, an attacker will force the MG to teardown Alice's conversation with Bob.

Table 2. Megaco Commands Which Can Disrupt

ServiceChange Restart	causes MG to hard reboot. Used in registration initial set up.
Notify Event OnHook	signifies called party has hung up
Subtract Termination	deletes ephemeral termination created for media exchange
Modify Media SendOnly	changes media protocol to send but not receive voice packets

Once an attacker knows which packet corresponds to which message, another attack is selective dropping of Alice's important packets. For example, an attacker can prevent Alice's call from even getting set-up by detecting her *Notify event 'Dial Digits'* and dropping it. Table 3 lists three ways in which an attacker can identify the specific messages from MGC-MG communication.

Table 3. Methods for Identifying Specific Messages

Timing	*ServiceChange Restart* is the first message sent by MGC to MG after MG registration
Visualization	associate messages with observed physical events (*OnHook* command is observed after Alice hangs up phone)
Network Traffic Analysis	commands identified by traffic pattern; (*Subtract Ephemeral Termination* command sent when RTP media traffic of constant–sized packets stops)

Solution 1: Commands between the MGC and the MG are grouped into Transactions, each of which is identified by a TransactionID. Transactions consist of one or more actions and are presented as

TransactionRequests. Corresponding responses to a TransactionRequest are received in a single reply. Ordering of Transactions is not guaranteed - transactions may be executed in any order or simultaneously. Transactions are identified by a TransactionID, which is assigned by sender and is unique within the scope of the sender. The MG should explicitly check TransactionIDs have not been repeated. If a repeated command is found with a TransactionID then it should be silently discarded. TransactionID sequences should start from a unique random value for each session so that it is hard for an attacker to reuse messages from old sessions.

The MG should use the Megaco message-piggybacking feature to make it difficult for an attacker to correlate Alice's particular messages based on visualization and network traffic analysis. Multiple commands can be sent simultaneously by the MG to the MGC randomly shuffling their order in the message body. MGs can also be made less predictable by adding random delays between the occurrence of a physical event (hanging up) and its notification to the MGC.

Problem 2: An attacker may degrade VoIP QoS. An attacker may exploit the property of VoIP media communication that it is highly sensitive to delay and jitter. Delays greater than 150 ms cause a conversation to become uncomfortable. This level of delay is usually the point at which both parties begin to speak at the same time. Jitter manifests to the listener as pops/clicks, words missing, and/or garbled speech. If jitter values exceed 50ms it is usually considered poor voice quality,

To reduce the impact of jitter, VoIP phones usually have a jitter buffer. The jitter buffer is usually designed to hold 1-2 datagrams and may adjust dynamically based on the perceived jitter. As datagrams arrive, they are placed in the jitter buffer, which holds them long enough to supply them to the codec at a constant rate. If a datagram arrives too early or too late, it may not fit in the jitter buffer and is discarded. Ideally the jitter buffer should be just large enough to handle the maximum delay variation, however, for every millisecond that you increase the jitter buffer you also add a millisecond of delay.

The attacker has two options: (1) faking some of Alice's packets by changing the SSRC field, which designates the source of RTP packets (to impersonate Alice) and inject artificial packets with higher sequence numbers that will cause the injected packets to be played in place of the real packets; or (2) sending garbage packets meaning both the header and the payload are filled with random bytes corrupting Bob's jitter buffer (most likely this will cause Bob's VoIP phone to crash or cause exorbitant delay processing which disrupts established VoIP conversations).

Solution 2: A VoIP implementation should have an intrusion detection system (IDS) or firewall on the phone itself that checks the media packet flow. Either the IDS or firewall can then be used to ensure: (a) packets with very large sequence numbers are discarded, (b) garbage packets are identified and discarded before the codecs try to mathematically decode them, and (c) large traffic volume for a single RTP flow signal a warning.

Problem 3: An attacker can mount a DoS attack on the VoIP signaling. An attacker can prematurely tear down Alice's and Bob's direct SIP signaling connection by sending a fake *SIP BYE* or *ICMP Port Unreachable* message to either of the parties. This results in the attacked party stopping the media transmission immediately while the other party continuing the conversation oblivious that the connection has been degraded or terminated.

Solution 3: In order to detect this attack, the VoIP infrastructure should create a rule that it detects orphan RTP flows. 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.

Consequences: This design patterns has the advantage of users having increased security for their VoIP calls but the disadvantage of increased delay.

C. VoIP Security Design Pattern Three

Securing VoIP against Eavesdropping

Example: RTP is not a complete protocol but a rather a framework where vendors are provided implementation freedom according to their specific application profiles. RTP facilitates network transport functions for real-time data, provides application level framing, and usually runs over an unreliable transport protocol such as UDP that does not guarantee the timely delivery of packets in order. RTP is usually implemented in conjunction with higher layer control that provides feedback information on transmission quality.

Problem: Security facilities provided by RTP alone are inadequate. RTP provides a framework for implementing high-level protocols which in turn can implement their own security services that may eventually provide benefit to RTP itself (RFC 1889). RTP cannot rely on the underlying network just because it is transmitted over IP and considering IPSec will consequently provide the security services. Services provided by IPSec are not useful for protocols other than IP and also do not support multicast sessions. RTP is network independent and could use other protocols like ATM/AAL5 for transmission of real-time data for which IPSec is not relevant. We discuss a technique to provide actual RTP packet encryption which provides

reasonable privacy from eavesdroppers and does not change the RTP properties of out-of-order packets, delay, and jitter requirements.

Solution: The DES cryptographic algorithm is used in the DES-CBC mode. The DES-CBC mode has a random access property that guarantees lost packets only prevent decoding of themselves and the following packets of their specific blocks without affecting the remaining transmission. The overhead of DES is much lower than that of compression algorithms used by RTP. The slight disadvantage of this approach is that DES is typically implemented in hardware and difficult to implement in software. This makes the solution slightly harder to implement since most VoIP products are software based. Other than DES, the current AES standard encryption algorithm may also be used with Counter Mode and f-8 mode (normally used for wireless transmission). AES overcomes the flaws of individual bit manipulation introduced by the CBC mode since AES encryption/decryption of one packet does not depend upon preceding packets. It also provides increased security with a larger block size (128 bits) and larger encryption keys.

IV. SUMMARY

As VoIP market penetration increases there are growing security concerns. The 2005 NIST report is currently the most comprehensive source on VoIP security and in this role is a good summary for managers considering the potential use of VoIP, however, it does not provide practical details about specific solutions that can be emulated.

This is where design patterns come in – design patterns are accepted solutions to well known problems cataloged by and for the actual developers who have been working with the internals of the technology. Design patterns reveal insights on how software solutions are actually implemented in real systems. We review the general guidance within the NIST report as a vehicle to contrast it with the specific guidance of security software design patterns.

With this paper we contribute to the cataloging of VoIP security design patterns by presenting three patterns. We invite feedback on this work and contributions of other VoIP security design patterns using the forum in [16].

REFERENCES

[1] R. Barbieri, D. Bruschi, and E Rosti, "Voice over IPsec: Analysis and Solutions". *18th Annual Computer Security Applications Conference (ACSAC)*, 2002.

[2] S. A. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," *IEEE Infocom*, 2006.

[3] B. Blakley, C. Heath, and members of The Open Group Security Forum, "Technical Guide: Security Design Patterns," *The Open Group,* April 2004.

[4] C-N. Chuah, "Providing End-to-End QoS for IP based Latency sensitive Applications." *Technical Report, Dept. of ECE, University of California at Berkeley,* 2000.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[6] B. Goode, "Voice Over Internet Protocol (VoIP)". *Proceedings of the IEEE*, Vol. 90 No. 9, Sept. 2002.

[7] ITU-T, "Recommendation G.114 - One-Way Transmission Time," Feb 2003.

[8] ITU-T, "H.323 - Packet-Based Multimedia Communications Systems", Feb. 1998.

[9] JDSoft, "eNAT for MSN Messenger," <http://www.easyfp.com/>

[10] D. R. Kuhn, T.J. Walsh, and S. Fries, *Security Considerations for Voice Over IP Systems*, U.S. National Institute of Standards and Technology (NIST) Special Publication 800-58, January 2005.

[11] Qovia, Inc. <http://www.qovia.com/>

[12] L. Rising (editor), *Design Patterns in Communications Software,* Cambridge University Press, 2001.

[13] J. Rosenberg., "Traversal Using Relay NAT (TURN)", *Internet-Draft draft-rosenberg-midcom-turn-07*, Feb. 2005.

[14] J. Rosenberg, "Interactive Connectivity Establishment *(ICE), Internet-Draft draft-ietf-mmusic-ice-04*, Feb 2005.

[15] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering,* Wiley, 2006.

[16] *Security Patterns Homepage* <http://www.securitypatterns.org/>

[17] *Skype*, "Skype-The Whole World can Talk for Free," <http://www.skype.com>

[18] B. Smaalders, "Performance Anti-Patterns," *ACM Queue*, February 2006.

[19] *Voice over IP Security Alliance.* <http://www.voipsa.org/>

[20] K. Werbach, "Using VoIP to Compete," *Harvard Business Review*, September 2005.