Real-time Steganography with RTP

I)ruid

<druid@caughq.org>

http://druid.caughq.org

Who am I?

- Founder, Computer Academic Underground (CAU)
- Co-Founder, Austin Hackers Association (AHA!)
- Employed by TippingPoint DVLabs performing VoIP security research











Overview

- VoIP, RTP, and Audio Steganography
- Previous Research
- Real-Time Steganography
 - W Using steganography with RTP
 - Rroblems and Challenges
- **SteganRTP**
 - About, Goals, Etc.
 - Architecture, Operational Flow
 - Message Structures
 - ⊠ Functional Subsystems
- Live Demo
- Conclusions, Future Work
- ₩Q&A





VoIP? RTP?

Real-time Transport Protocol

Subsect Used by most VoIP systems to transmit call audio data





Audio Steganography

In 6 slides or less...

Steganography?

- Steganos (covered) graphein (writing)
- Hiding a secret message within a covermedium in such a way that others can not discern the presence of the hidden message
- Hiding one piece of data within another





Steganography Terms

- Message The data to be hidden or extracted
- Cover-Medium The medium in which information is to be hidden. Also sometimes called "cover-image/data/etc."
- Stego-Medium A medium within which information is hidden
- Redundant Bits Bits of data in a cover-medium that can be modified without compromising that medium's perceptible integrity





Types of Covert Channels

Storage-based

- **Persistent**
- Embedding message data into a static cover-medium
- Extracting message data from a static stego-medium

- **Transient**
- Signals message data by modulating behavior
- Extracts message data by observing effects of modulation





Digitally Embedding

- Digitally embedding a message in a covermedium usually involves two steps:
 - Identify the redundant bits of a cover-medium
 - Deciding which redundant bits to use and then modifying them
- Generally, redundant bits are likely to be the least-significant bit(s) of each data word value of the cover-medium





Digitally Embedding in Audio

- Audio is a very inaccurate type of data
- Slight changes will be indistinguishable from the original to the human ear
- In Audio, you can use the least-significant bits of each word value as redundant bits
- Use the redundant bits to minimize the impact of changes





Example: 8-bit Audio Embedding

Let's assume an 8-bit cover-audio file has the following 8 bytes of data in it:

0xb4, 0xe5, 0x8b, 0xac, 0xd1, 0x97, 0x15, 0x68

In binary:

10110100-11100101-10001011-10101100 11010001-10010111-00010101-01101000

We wanted to hide the byte value '214' (11010110), we replace the least significant bit from each byte to hide our message byte:

10110101-11100101-10001010-10101101 11010000-10010111-00010101-01101000

The modifications result in the following:

Original: 0xb4, 0xe5, 0x8b, 0xac, 0xd1, 0x97, 0x15, 0x68 Modified: 0xb5, 0xe5, 0x8a, 0xad, 0xd0, 0x97, 0x15, 0x68





Previous Research

Audio Steganography

🔀 Data Stash: MP3 files

http://www.skyjuicesoftware.com/software/ds_info.html

Hide4PGP: WAV and VOC files

http://www.heinz-repp.onlinehome.de/Hide4PGP.htm

InvisibleSecrets: WAV files

This is the secret state of the secret state o

MP3Stego: MP3 files

http://www.petitcolas.net/fabien/steganography/mp3stego/

ScramDisk: WAV files

Http://www.scramdisk.clara.net/

S-Tools 4: Embedding into a WAV file

The state of the s

Steganos: WAV and VOC files

ftp://ftp.hacktic.nl/pub/crypto/steganographic/steganos3r5.zip

StegHide: WAV and AU files

This is the standard of the st

StegMark: MIDI, WAV, AVI, MPEG

Thttp://www.datamark.com.sg/onlinedemo/stegmark/





VoIP Steganography

- A few previous research efforts
- Subsestion "Steganography":
 - W Using redundant bits to widen RTP audio band
 - **W** Using redundant bits for error correction
 - Replacing RTCP
 - Watermarking audio for integrity checking

Deficiencies:

- Some are just "theory" papers, don't explain how they intend to accomplish certain tasks
- ☑ Don't achieve the primary goal of steganography:
 - We use of steganographic techniques easily identifiable by an observer
 - Message data is trivially recognized and extracted from stego-medium
- Margin Only one public PoC; no full implementations
- Analysis paper forthcoming





Real-time Steganography

Or, utilizing steganographic techniques with an active network communications channel

Context Terminology

- Packet A network data packet
- Message Data being embedded or extracted via steganographic techniques





"Real-time" Steganography?

- Separate "hide" and "retrieve" modes are common in storage-based steganography implementations
- Common cover-mediums are static or unidirectional
- ₩What about Vo²IP?
- Utilizing steganography with RTP provides the opportunity to establish an active, or "real-time" covert communications channel





RTP's Redundant Bits

- RTP packet payloads are encoded multimedia
- I'll be focusing on RTP audio
- RTP supports many different audio Codecs
- RTP's redundant bits are determined by the codec used
- 8-bit sample size Codecs are generally resilient to changes of the LSB for each sample
- Larger sample size Codecs may provide for one or more LSBs to be modified per sample











Audio Codec Word Sizes

G.711 alaw: 8-bit word size

G.711 ulaw: 8-bit word size

Speex: dynamic, variable word size

ZiLBC: class-based bit distribution





Throughput

☑ G.711 (ulaw/alaw):

- 160 byte RTP payload
- 8-bit sample word size

- $\mathbb{Z}(160/8)*50 == 1,000 \text{ bytes/sec}$





Problems and Challenges

Trying to use steganography with RTP

Unreliable Transport

Problems:

- RTP uses UDP as it's transport protocol
- Market UDP is connectionless and unreliable

- Data split across multiple packets may arrive out of order
- One or more parts of data split across multiple packets may not arrive at all





Cover-Medium Size Limitations

Problems:

- Individual RTP packets don't provide much space for embedding message data
- March Different audio Codecs use different audio word sizes

Large message data will likely be split across multiple packets and will need to be reassembled





Latency

Problems:

RTP is extremely sensitive to network latency and other QoS issues

Challenges:

- Overall system must not interfere too much with RTP packet routing
- Use of steganography cannot delay any individual RTP packet for too long





RTP Streams

Problems:

RTP employs two separate half-duplex packet streams to achieve full-duplex communication

Both RTP streams must be correlated and tracked for an individual session





Compressed Audio

Problems:

May Audio being transferred by RTP may be compressed

Challenges:

- Identification of compressed audio
- Packets containing compressed audio must either
 - Not be used
 - Be decompressed, modified, and then recompressed in order to embed message data





Media Gateway Audio Modifications

Problems:

Intermediary media gateways may re-encode audio, change the codec entirely, or otherwise modify the RTP audio payload

- Identification of intermediary media gateway interference
- Overcome the particular type of interference





Audio Codec Switching

Problems:

Endpoints may switch audio Codecs mid-session

- ☑ Identifying a change in audio codec
- Creating an adaptable steganographic embedding method





SteganRTP

My reference implementation.

About SteganRTP

- Most awesome tool name I've ever created
- Linux application
- Windowed curses interface
- Must be able to modify the outbound RTP stream's packets
- Must be able to observe the inbound RTP stream's packets
- Pair with ARP poisoning for active MITM





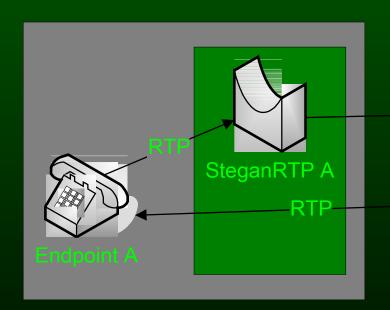
Goals

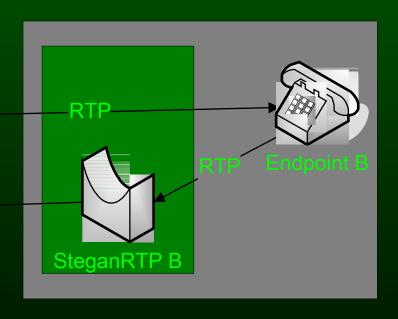
- Steganography: Hide the fact that covert communication is taking place
- Full-Duplex Communications Channel
- Compensate for unreliable transport
- Transparent operation whether hooking locally generated/destined packets vs. forwarded packets
- Simultaneous transfer of multiple types of data





Architecture: Endpoint

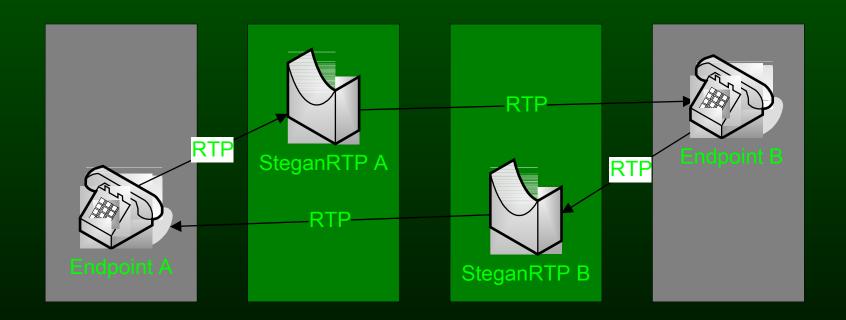








Architecture: MITM







Process Flow







Identify RTP Session

- Using libfindrtp, one of my previous projects
- Identifies RTP sessions between two endpoints
- Identifies RTP during call setup by observing VoIP signaling traffic
- Supports RTP session identification via SIP and Skinny signaling protocols





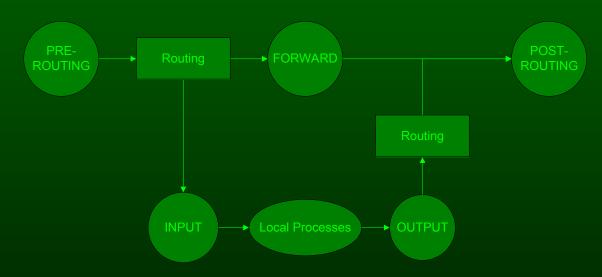
Hooking Packets

- Linux NetFilter Hook Points
 - Basically, an iptables rule with target QUEUE
- NetFilter User-space Queuing Agent
 - API for reading, writing, or passing packets destined for the QUEUE target





Linux NetFilter Hook Points



- Anywhere you can insert an iptables rule:
- Locally Originated or Destined:
 - **INPUT**
 - **W**OUTPUT
- Packet Forwarding:
 - **FORWARD**
- NAT, SNAT, etc:
 - **PREROUTING**
 - POSTROUTING





Hooking Packets

- SteganRTP registers itself as a user-space queuing agent for NetFilter via libipq
- SteganRTP creates two rules in the NetFilter engine with targets of QUEUE:
 - Matching the Inbound RTP stream at PREROUTING
 - Matching the Outbound RTP stream at POSTROUTING

SteganRTP is then able to:

- Read packets from the queue
- Modify them if needed
- Place them back into the queue
- Tell the queue to accept the packet for further routing





Inbound Packets

- Immediately accept the packet for routing
- Extract the message
- Decrypt the message
- Verify message's checksum
- Send message to the message handler





Outbound Packets

- Poll for data waiting to go out
 - If there isn't any, immediately forward the RTP packet unmodified
- Create a new message with header based on properties of the RTP packet
- Read as much of the waiting data as will fit in the message
- Encrypt the message
- Embed the message into the RTP payload covermedium
- Send the RTP packet



Session Timeout

- If no RTP packets are seen for the timeout period, all session information is dropped
- Control returns to libfindrtp, which searches for a new session





Message Handler

- Receives all valid incoming messages
- Performs internal state changes and administrative tasks in response to control messages such as:
 - Echo Request
 - Echo Reply
 - Resend of lost messages
 - Prep for receiving a file
 - ⊠ Closing a finished file
- Receives incoming user chat data
- Receives incoming file data
- Receives incoming shell data





Packets and Messages

Yay bits!

RTP Packet Format

```
RTP Header:
  |V=2|P|X|
             |M|
                              sequence number
                      timestamp
          synchronization source (SSRC) identifier
  contributing source (CSRC) identifiers
                  Encoded Audio Data
```





Message Format

Header:





Message Header Fields

- ☑ID (32 bits):
- Seq (16 bits):
 - Message Sequence Number
- Type (8 bits):
 - Message Type
- Length (8 bits):
 - Length of remaining message data





Message Types

□ 0: Reserved

₹1: Control

₹10: Chat Data





Message Type: Control





Control Types

□ 0: Reserved

2: Echo Reply

3: Resend

₹5: End File





Control Message: Echo Request





Control Message: Echo Reply





Control Message: Resend





Control Message: Start File

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 4 5 6 7 8 9 0 1 5 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 6 7 8 9 0 1 1 6 7 8 9 0 1 1 6 7 8 9 1 8 9 0 1 1 6 7 8 9 1 8 9 0 1 1 6 7 8 9 1 8 9 0 1 1 6 7 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9 1 8 9
```





Control Message: End File





Message Type: Chat Data





Message Type: File Data





Message Type: Shell Data





Functional Subsystems

The parts that make it go.

Encryption System

- Light-weight, pseudo-encryption (XOR)
- Could be replaced with real crypto if no impact on RTP stream latency
- XOR pad is a SHA1 hash of a shared secret
- XOR operation is begun at an offset into the hash
- keyhash:
 - Sha1(shared-secret)
- keyhash_offset
 - RTP_TS) % 20





Embedding System

- Currently supports G.711
- Use common LSB embedding method
- Properties of the RTP packet determine a total available size for embedding
- - RTPPayloadSize / (wordsize * 8)
- - Available MessageHeaderLen





Extracting System

- A reverse of the Embedding function
- Then a pass through the crypto function
- Verification of the ID field checksum





Outbound Data Polling System

- Linked list of file descriptors that may have data waiting to go out:
 - RAW message interface

 - Chat data
 - Input for Remote Shell service
 - Output from Local Shell service (if enabled)
 - Individual File transfer data
 - 聚...
- Prioritized in the above order





Message Caching System

- All inbound and outbound messages are cached
- If the remote app requests a resend, it is read from the cache and written to the RAW message interface
- If the local app receives future messages, they are available in the cache once the correct expected message is received





Challenges Met

How SteganRTP addresses the Problems and Challenges identified earlier

Unreliable Transport

- Request and identification of resent messages
- Re-ordering out of order messages
- Identifies un-requested, replayed messages to provide replay protection (bonus!)





Cover-Medium Size Limitations

- Plenty of RTP packets being sent per second
- User data can be spread over multiple messages and packets and then reassembled
- An achieved throughput of 1000 bytes per second is functional for my purposes
- (not adequate for transferring your massive pr0n collection)





Latency

- RTP packets can be "skipped" and sent along unmodified
- Fast pseudo-cryptography (XOR!) is used rather than full cryptography
- Crypto only needs to provide obfuscation entropy prior to embedding the individual bits, not protect the data





RTP Streams

- Elibfindrtp for identification
- Elibipq for tracking and hooking packets





Audio Codec Switching

- Embedding parameters are derived from RTP packet properties
- Each RTP packet is processed individually
- If an audio codec isn't supported, the packet is passed unmodified

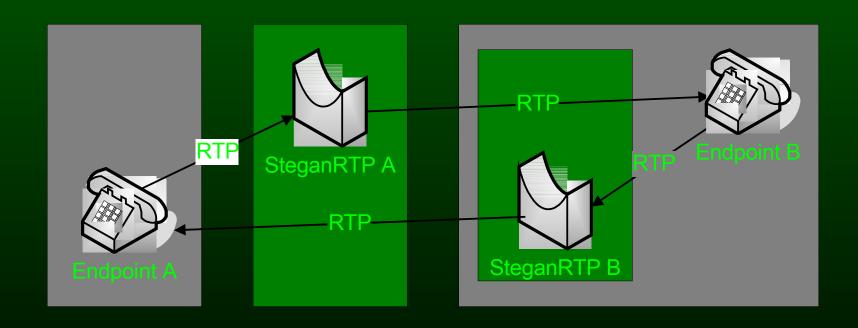




Live Demo!

Or, I)ruid likes to tempt fate...

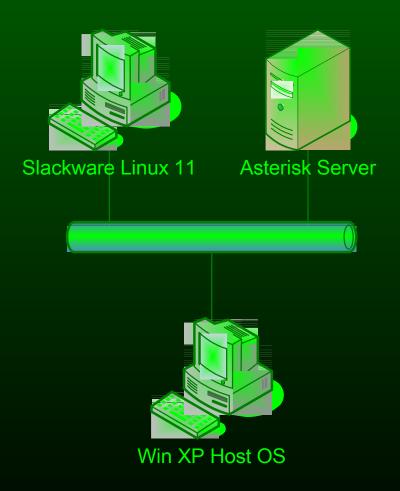
Demo Scenario







Demo Virtualized Environment







Conclusions

- Met all of my initial design goals
- Met most of the identified challenges

 - Media Gateway interference
- - Prevents the MITM scenario
 - Prevents the endpoint scenario in some cases





Future Work

- Improve G.711 codec's embedding algorithm
 Silence/Voice detection
- Create embedding algorithms for additional audio Codecs
- Create embedding algorithms for video Codecs
- Use real crypto instead of XOR
- Support for fragmenting larger messages across multiple RTP packets
- Expand Shell access functionality into a services framework
- White paper detailing research and implementation





Source Code

[™]SteganRTP

http://sourceforge.net/projects/steganrtp/

Blibfindrtp

http://sourceforge.net/projects/libfindrtp/





Q & A





References

- SteganRTP
 - Http://sourceforge.net/projects/steganrtp/
- **B**libfindrtp
 - http://sourceforge.net/projects/libfindrtp/
- Steganography Tools List
 - http://www.jjtc.com/mwiki/index.php?title=Main_Page
- RTP Specification
 - March http://www.ietf.org/rfc/rfc1889.txt
- RTP Parameters (Type/Codec values list)
 - Mhttp://www.iana.org/assignments/rtp-parameters



