# DisAsterisk Sneak-Peak

Leveraging Open Source Software for Vulnerability Research and Mayhem

I)ruid && intropy
Computer Academic Underground
TippingPoint, a division of 3Com

# Who da fuck are we?

- Me:
  - Founder, CAU
  - Co-Founder, AHA!
  - TippingPoint VoIP Security Researcher
  - VoIP vuln research
- Him:
  - Member, CAU
  - AHA!
  - TippingPoint Security Researcher
  - Reversing / broader-scoped vuln research
  - Punk bitch who wouldn’t come to Seattle

# What is Asterisk?

- Internet Protocol PBX (IP PBX) in software
- Open Source (GPL)
- Supports Many Signaling Protocols:
  - IAX™ (Inter-Asterisk Exchange)
  - H.323
  - SIP (Session Initiation Protocol)
  - MGCP (Media Gateway Control Protocol)
  - SCCP (Cisco® Skinny®)
- Supports Media Protocols:
  - IAX™ (Inter-Asterisk Exchange)
  - RTP (Real-time Transport Protocol)

## What is DisAsterisk?

- An *extremely* young project
  - (~12 hours of actual development so far)
- An Asterisk Module and Patch
- Has Multiple Independent Components
- Leverages Asterisk's existing functionality:
  - Full-featured CLI for user interaction
  - Protocol state machines
  - Protocol packet dissectors

## Thanks Beetle!

- “DisAsterisk”, what a fucking AWESOME project name!

## Asterisk Modules

- Compiled as a shared object (module.so)
- Dynamically Loaded:
  - Auto-loaded if placed in /usr/lib/asterisk
  - Manually loaded from the Asterisk CLI
- Uses a standard, documented Asterisk Module API
- Can register with the Asterisk CLI to provide module-specific commands to users

# Asterisk Module API

- Local Static Chars:
  - tdesc: Long Module Name
  - desc: Short Module Name
  - synopsis: Synopsis of Module Functionality
  - descrip: Long Module Description

- Local Functions:
  - load_module(): Module initialization, registeres CLI commands, etc.
  - unload_module(): Module cleanup, unregisters CLI commands, etc.
  - description(): Returns static char “desc”

## Asterisk Module CLI

- load_module() should register CLI commands via a call to ast_cli_register_multiple() with an ast_cli_entry structure:

```
struct ast_cli_entry {
    /* Null terminated list of the words of the command */
    char *cmda[AST_MAX_CMD_LEN];
    /* Handler for command (fd for output, # of args, arg list) */
    int (*handler)(int fd, int argc, char *argv[]);
    /* Summary of the command (<60 characters) */
    char *summary;
    /* Detailed usage information */
    char *usage;
    ...
}
```

# Example Module CLI: Fuzzer

```
static struct ast_cli_entry fuzz_cli[] = {
   { { “fuzzing”, NULL }, disast_cli_fuzzing, “Toggles Fuzzing
   Globaly”, disast_cli_fuzzing_usage, NULL },

   { { “fuzz”, “rtp”, NULL }, disast_cli_fuzz_rtp, “Toggles
   Fuzzing RTP”, disast_cli_fuzz_rtp_usage, NULL },

   { { “fuzz”, “rtp”, “header”, NULL }, disast_cli_fuzz_rtp,
   “Toggles Fuzzing RTP Header”, disast_cli_fuzz_rtp_usage, NULL }
   ,

   { { “fuzz”, “rtp”, “payload”, NULL }, disast_cli_fuzz_rtp,
   “Toggles Fuzzing RTP  Payload”, disast_cli_fuzz_rtp_usage, NULL
   },
};
```

## The DisAsterisk Module

- disasterisk.so
- On-Load:
  - Initializes module data structures
  - Registers CLI commands
- All DisAsterisk components begin in a dormant state
- CLI controls all DisAsterisk functionality
- This is the majority of the code

## The DisAsterisk Patch

- Stuff we couldn’t do within the context of the module and API
- Right now it’s 2 lines of code.

## Current DisAsterisk Components

Existing Now...

# Protocol Fuzzer

- Protocols:
  - Current Protocols:
    - RTP (Real-time Transport Protocol)
  - Upcoming Protocols:
    - IAX™ (Inter-Asterisk Exchange)
    - H.323
    - SIP (Session Initiation Protocol)
    - MGCP (Media Gateway Control Protocol
    - SCCP (Cisco® Skinny®)
- Uses our own simple fuzzing logic for value selection

## Protocol Fuzzer

- Requires a small patch to Asterisk in addition to the module to:
  - Hook decoded/parsed packets
  - Dispatch them to the module
- Allows us to easily modify specific aspects of the packet for granular and selective intelligent fuzzing

# Protocol Fuzzer Commands

- fuzzing [on | off | status]
  - Enable, disable, or verify global fuzzing status
- fuzz [protocol] <args>
  - on: Enable fuzzing of protocol
  - off: Disable fuzzing of protocol
  - status: Verify status of fuzzing for protocol
- fuzz rtp <args>
  - header <args>
    - version [off | random | seq]: Version field
    - timestamp [off | random | seq]: Timestamp field
    - ...
  - payload [off | random | seq]

## Upcoming DisAsterisk Components

Things yet to be implemented...

## VoIP Scanner

- Scanner Targets:
  - Registration Server username enumeration
  - Network Map / Port Map
  - Endpoint capabilities

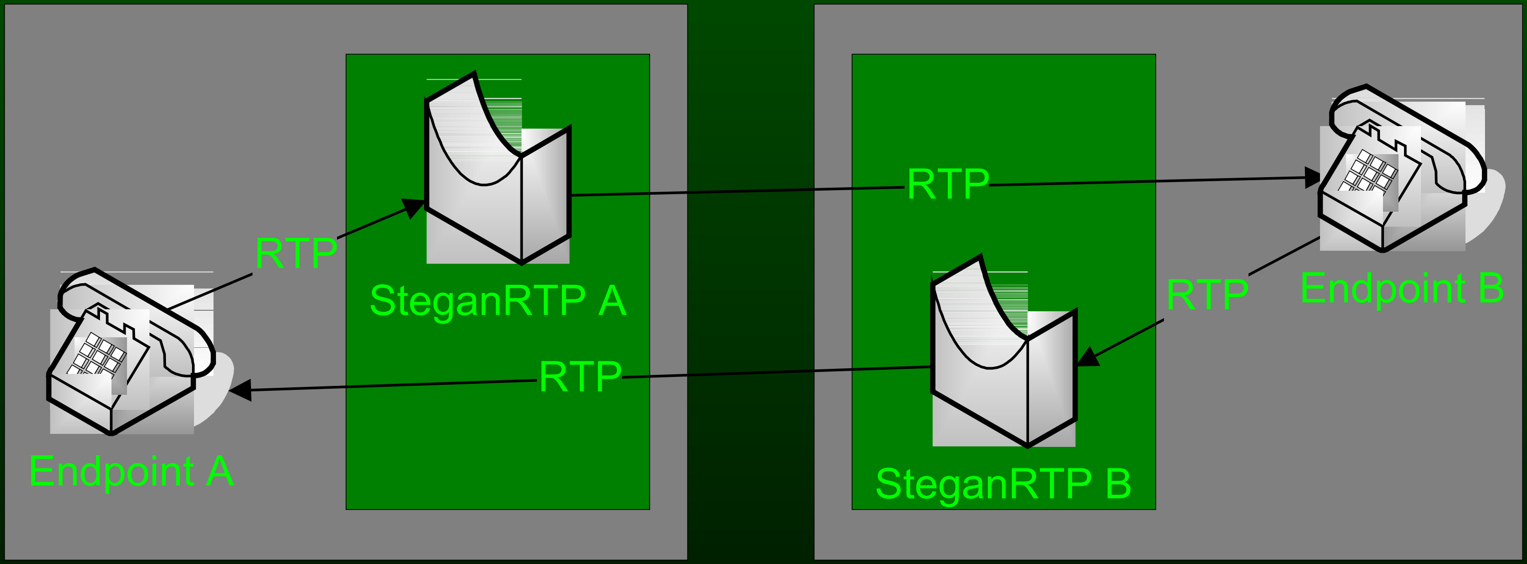
## SteganRTP

- Covert channel within the audio payload of an RTP stream

## SteganRTP Goals

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

# SteganRTP Architecture



## Embedding Message into Cover

- XOR entire steg packet against keyhash starting from keyhash[keyhash_offset]
- Use common LSB embedding method
- Embed entire steg packet into cover medium (RTP payload)

# Steg Packet Format

Header:

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Checksum / ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Sequence            |     Type      |    Length     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Packet Body:

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Value (Type-Defined Body)                    |
!                                                               !
.                                                               .
```

## Important Values

- keyhash:
  - sha1(shared-secret)
- keyhash_offset
  - hashword( keyhash, RTP_Seq, RTP_TS ) % 20
- Available:
  - RTPPayloadSize / (wordsize * 8)
- MessageSize:
  - Available - StegHeaderLen

## Steg Packet Header Fields

- ID (32 bits):
  - hashword( keyhash, (Seq + Type + Len) )
- Seq (16 bits):
  - Packet Sequence Number
- Type (8 bits):
  - Packet Type
- Length (8 bits):
  - How many bytes used (up to “Coverlen”)

## Feedback!

What we’re wanting from YOU!

## We’re looking for:

- More DisAsterisk component ideas!
  - What would you want DisAsterisk to do?
- A good C fuzzing library for value selection & value state tracking
  - We’ve only implemented very basic value selections
- A name for my steg protocol that I developed for SteganRTP
  - SRTP already taken (Secure RTP)

Questions?
Comments?
Feedback?

## Fin

I)ruid

<druid@caughq.org>

http://druid.caughq.org

intropy

<intropy@caughq.org>

http://www.caughq.org/~intropy/