COVERT TCP WITH A TWIST

DEF-CON 2017 Wall of Sheep

Mike Raggo and Chet Hosmer

INTRODUCTION

Chet Hosmer *is an international author, educator and researcher and founder of Python Forensics, Inc. a Non-Profit Research Institute focused on the collaborative development of open source investigative technologies using the Python programming language.*

- A Visiting Professor at Utica College in the Cybersecurity graduate program, where his research and teaching is focused on data hiding, active cyber defense and security of industrial control systems.

- Adjunct Professor at Champlain College in the Digital Forensics Graduate Program, where his research and teaching is focused on solving hard digital investigation problems using the Python programming language.

*Chet is also a co-*founder of WetStone Technologies

cdh @ python-forensics.org

INTRODUCTION

Mike Raggo *is Chief Security Officer at 802 Secure and has over 20 years of security research experience.*

His current focus is wireless IoT threats impacting the enterprise. Michael is the author of "Mobile Data Loss: Threats & Countermeasures" and "Data Hiding: Exposing Concealed Data in Multimedia, Operating Systems, Mobile Devices and Network Protocols" for Syngress Books, and contributing author for "Information Security the Complete Reference 2nd Edition".

A former security trainer, Michael has briefed international defense agencies including the FBI and Pentagon, and is a frequent presenter at security conferences, including Black Hat, DEF CON, Gartner, RSA, DoD Cyber Crime, OWASP, HackCon, and SANS.

@DataHiding

THE ORIGINS

Covert Channels in the TCP/IP Protocol Suite

by Craig H. Rowland

May 5, 1997

http://firstmonday.org/ojs/index.php/fm/article/view/528/449

COVERT COMMUNICATIONS

". . . any communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy…."

Source: U.S. Department of Defense. Trusted Computer System Evaluation "The Orange Book". Publication DoD 5200.28-STD. Washington: GPO 1985

Covert Channels in the TCP/IP Protocol Suite

by Craig H. Rowland

May 5, 1997

Method 1:

Manipulation of the IP Identification Field

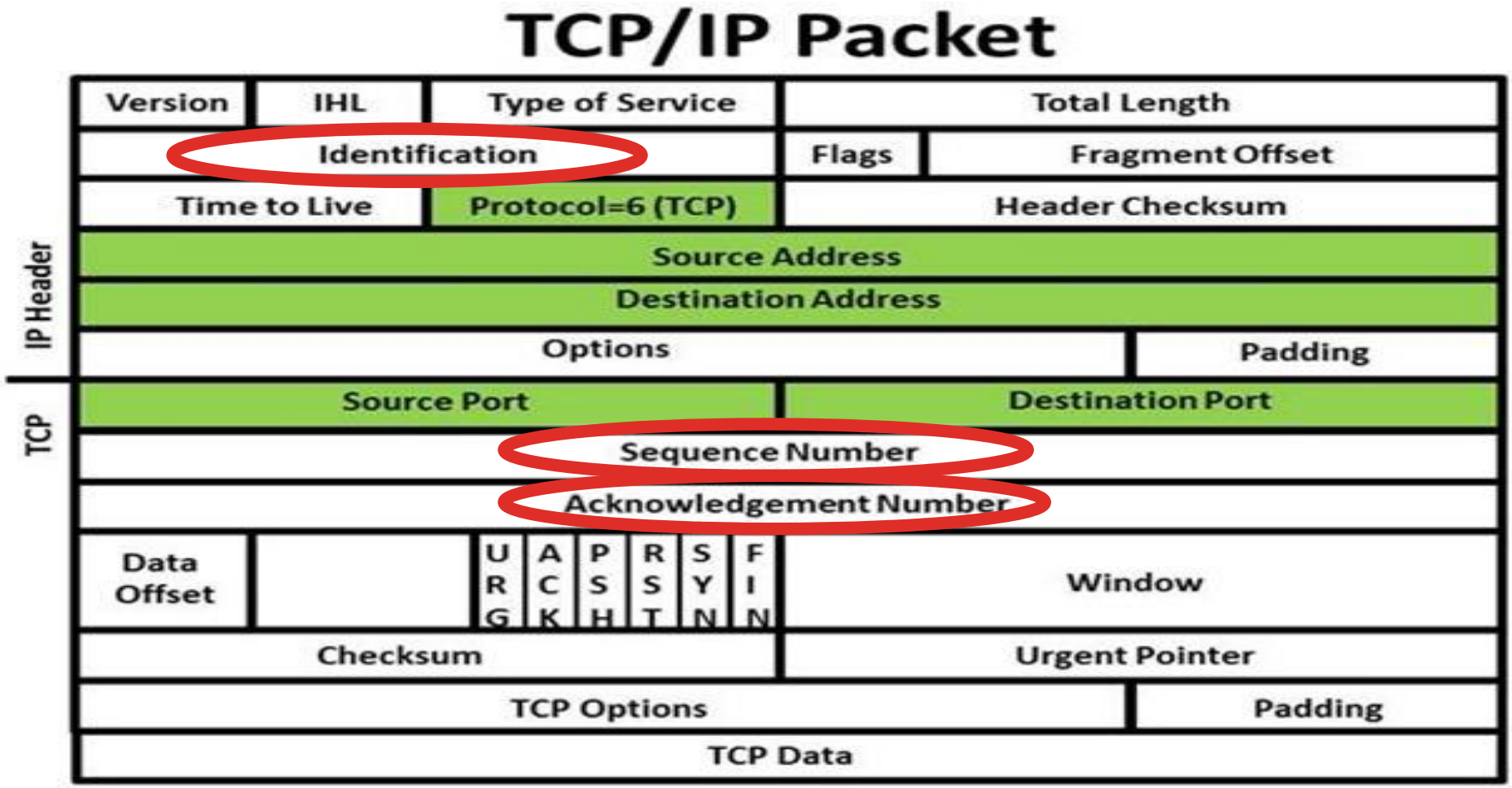
Method 2:

Manipulation of the Initial Sequence Number

Method 3:

Manipulation of the TCP Acknowledge Sequence Number Field “Bounce”

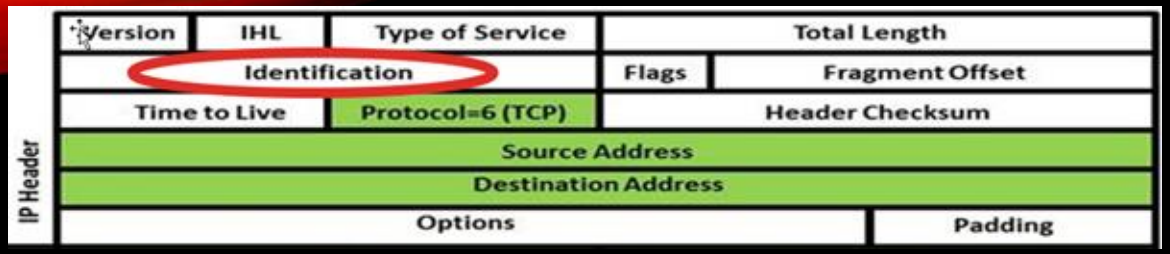
QUICK REVIEW OF TCP/IP HEADERS



BASIC CONCEPT

TCP/IP Packet Headers contain standard fields that must be present to comply with the protocol standards.

Several of these standard fields can be manipulated to carry hidden content.



METHOD ONE IP IDENTIFICATION FIELD

Overview of Method One

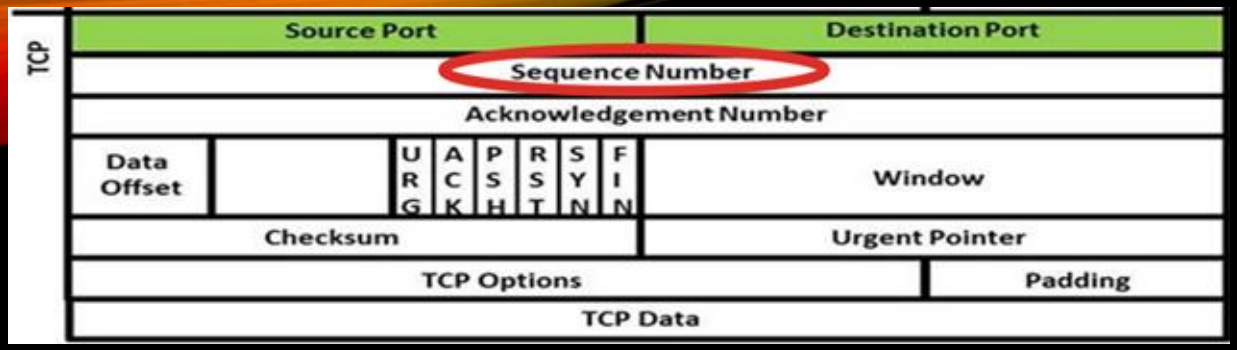
- This is by far the simplest method. The identification field of the IP Header gives each packet a unique identifier in order handle possible (but unlikely today) packet fragmentation.
- If the sender encodes this field with hidden content (encoded, encrypted or using plaintext) the receiver can collect data from the ID Field of each packet and then re-assemble them into a message.

METHOD ONE EXAMPLE

```
08:22:31.450099 192.168.0.122.5500 > 192.168.0.133
S 5400660022:5400660022(0) win 512 (ttl 64, id 17152)
```

Converting the ID field

1) The receiver would divide by a pre-defined constant.
2) In this case 256. 17152 / 256 = 67
3) Converting 67 into ASCII = the letter C



METHOD TWO INITIAL SEQUENCE NUMBER

- **Manipulation of the Initial Sequence Number Field***
  - The Initial Sequence Number is used to establish a communication link between a client and remote server
  - A program can be created to generate this number using a constant divided by an ASCII character value (or much larger values i.e. words, double words, quad words)
  - A similar program on the other end can passively listen for communication and then decode the message

METHOD TWO
INITIAL SEQUENCE NUMBER

TCPDUMP Packet Header

```
20:30:10.005553 10.1.1.0.45321 > 128.162.1.0.80:
S 1207959552:1207959552(0) win 512 (ttl 64, id 49408)
```

Time Stamp	Source IP/Port	Destination IP/Port
20:30:10.005553	10.1.1.0.45321	128.162.1.0.80

Flag	Sequence Number	Additional Data
S	1207959552:1207959552 (0)	win 512 (ttl 64, id 49408)

METHOD TWO
INITIAL SEQUENCE NUMBER

Locate ISN

1207959552

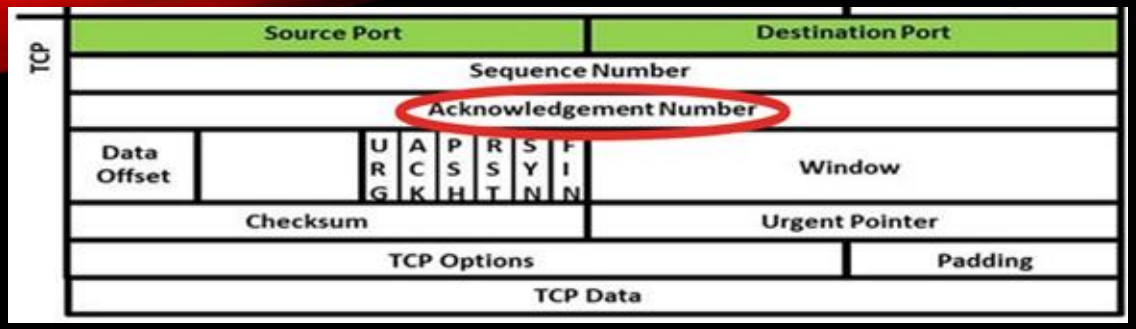
Divide by constant

1207959552 / 16777216 = 72

Convert to ASCII

72 = "**H**" in ASCII

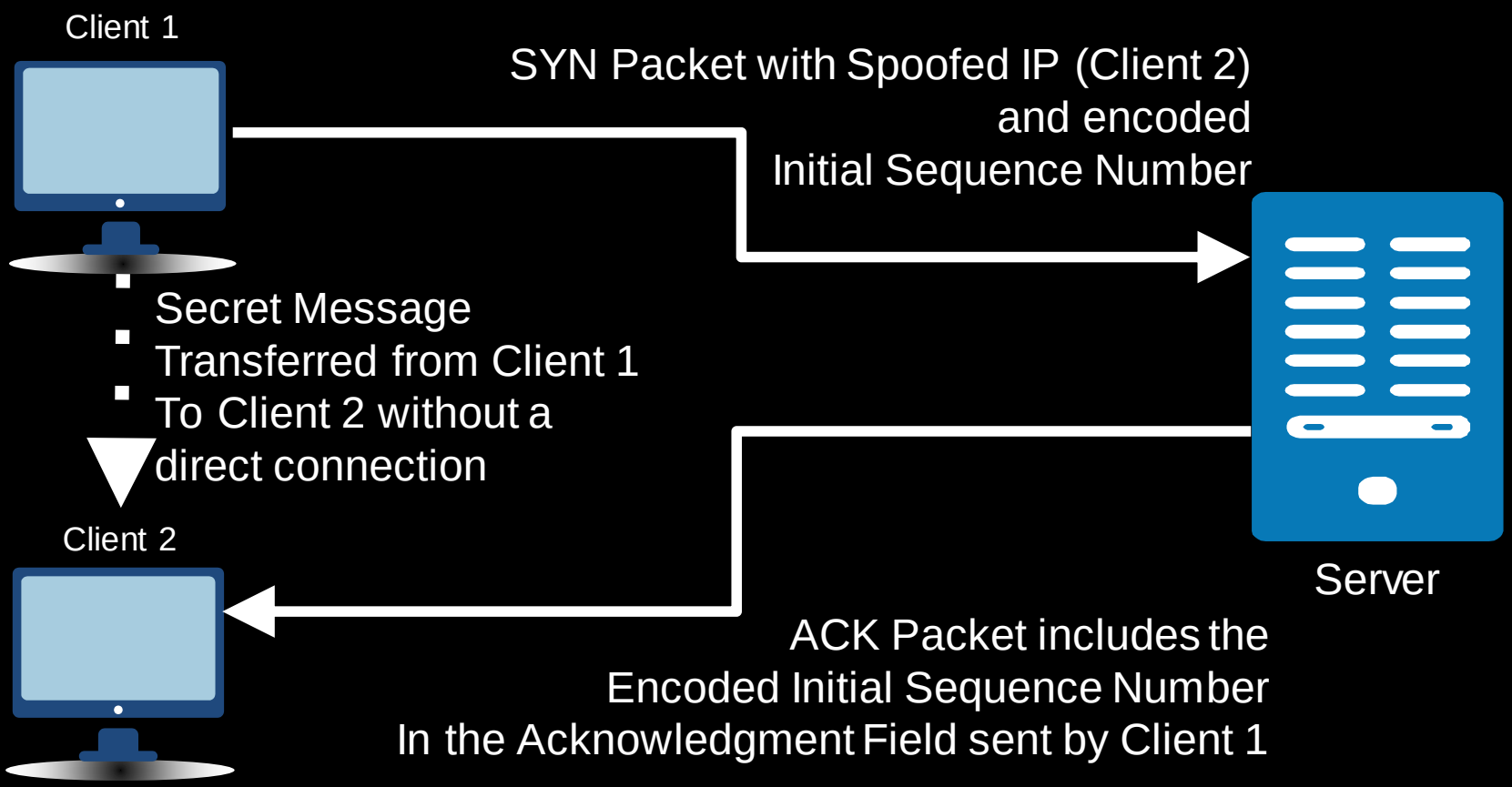
METHOD THREE TCP ACKNOWLEDGE SEQUENCE NUMBER FIELD "BOUNCE"



Overview of Method 3

1. Method 3 expands on Method 2 and adds IP Spoofing to conceal the sender identity.
2. This method enables the sender to remain autonomous by sending SYN packet with an initial sequence number to a server while spoofing the IP address of the sender.
3. The response to the SYN bounces of the target server and is directed to the spoofed IP address.
4. Allowing the sender of the original SYN to remain anonymous.

METHOD 3 ILLUSTRATION



MODERN DAY METHODS

Covert UDP (using IoT)

About IoT

- Many consumer, and even enterprise as well as “industrial things” have weak inherent built-in security
- Many devices support UPnP to allow an app or other devices to discover (M2M)
- Sends multicast packets broadcasted to local network

EXPLOITATION OF SSDP ULA OPT FIELD - UDP COVERT

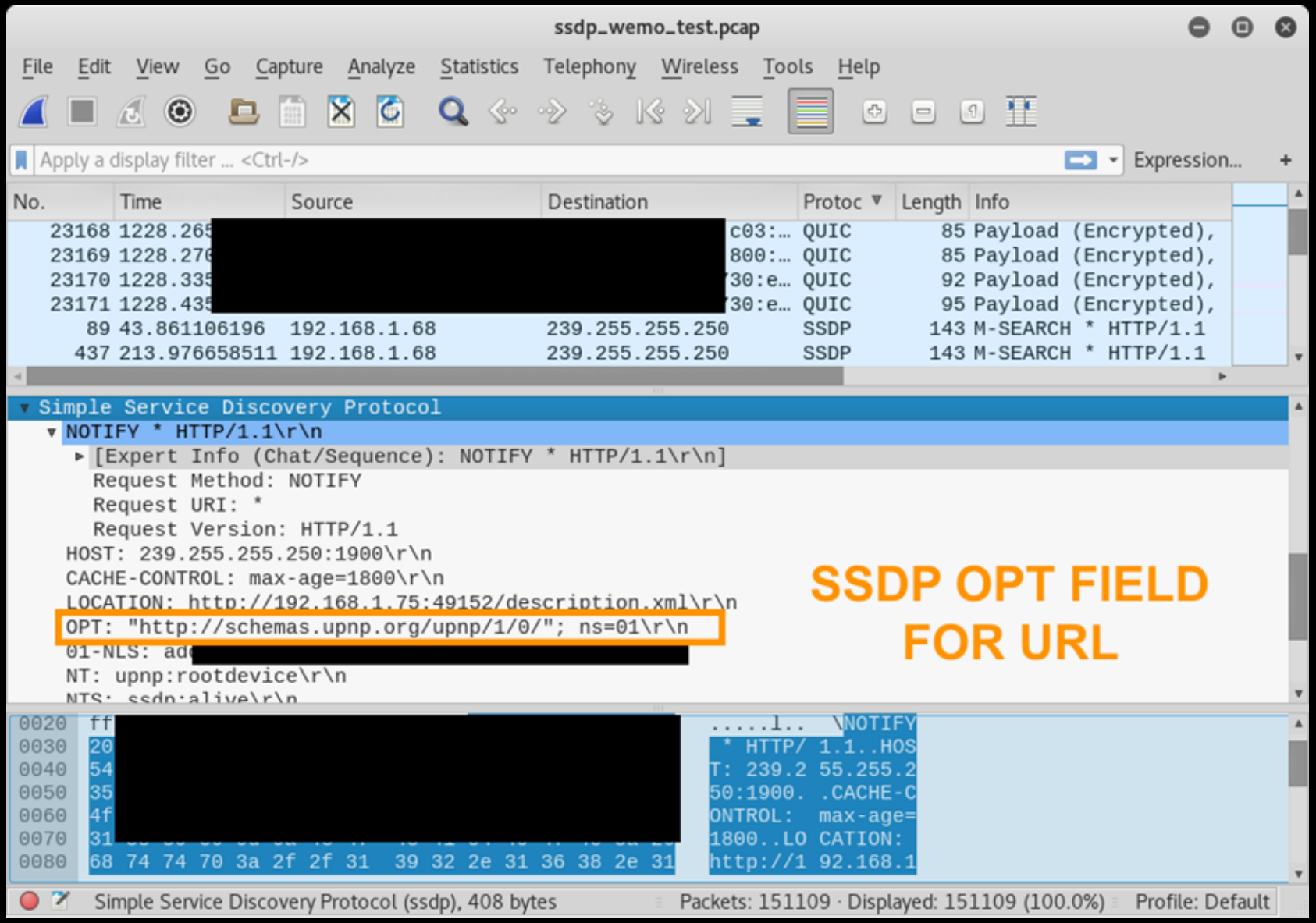
- **SSDP UPNP**
  - Simple Service Discovery Protocol (Part of Universal Plug and Play)
  - M-SEARCH - Discover packet sent by app or another device
  - NOTIFY - Device announces itself on the network, routinely, and also when it leaves

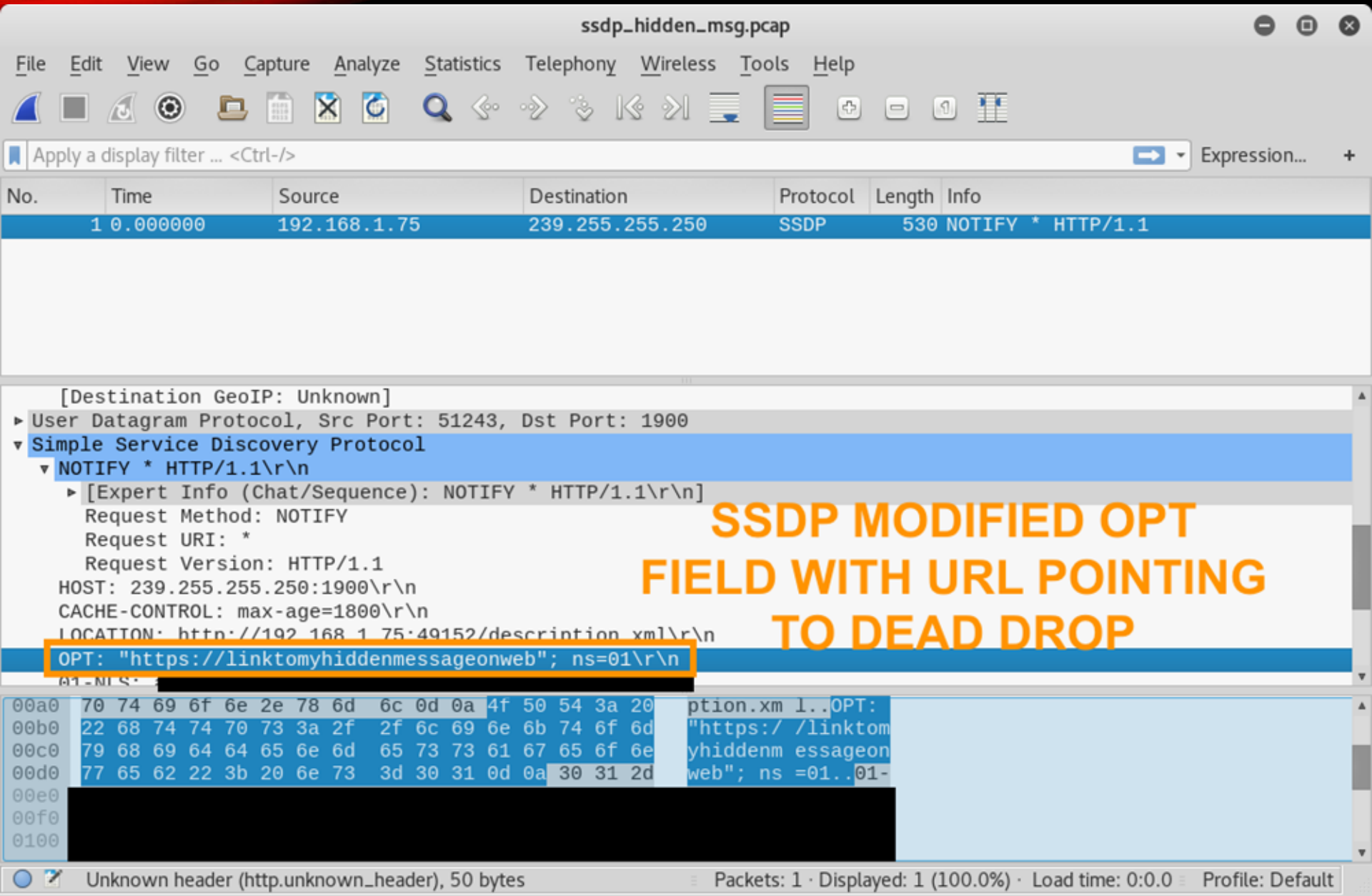
EXPLOITATION OF SSDP ULA OPT FIELD - UDP COVERT

- **ULA OPT FIELD**
  - Unique Local Addresses - Site-Routable
  - Used in NOTIFY and M-SEARCH messages
  - For use in IPv4 and IPv6 (for backward compatibility)

Reference: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1-AnnexA.pdf

UDP - EXPLOITATION OF SSDP





SSDP MODIFIED OPT FIELD WITH URL POINTING TO DEAD DROP

UDP - EXPLOITATION OF SSDP

```
▼ Simple Service Discovery Protocol
  ▼ NOTIFY * HTTP/1.1\r\n
    ▶ [Expert Info (Chat/Sequence): NOTIFY * HTTP/1.1\r\n]
      Request Method: NOTIFY
      Request URI: *
      Request Version: HTTP/1.1
    HOST: 239.255.255.250:1900\r\n
    CACHE-CONTROL: max-age=1800\r\n
    LOCATION: http://192.168.1.75:49152/description.xml\r\n
    OPT: "https://linktomyhiddenmessageonweb"; ns=01\r\n
```

- **MODIFY SSDP OPT FIELD WITH HIDDEN MESSAGE, URL, ETC.**
- **COVERT COMMUNICATIONS, DEAD DROP, MALWARE CALLBACK TO CNC FOR UPDATES, ETC**

EXPLOITATION OF SSDP ULA OPT FIELD - UDP COVERT

- **SSDP ULA OPT FIELD**
  - Designed for a URL of your choice, could be used to point to a site for the message, an image with a hidden message, CnC, etc.
  - Doesn’t have to be a URL - testing revealed it is essentially a free-form field
  - Can be combined with crypto (XOR, etc. to further protect the message)

EXPLOITATION OF SSDP ULA OPT FIELD - UDP COVERT

- **SSDP ULA OPT FIELD**
  - SSDP in general is designed ideally for the internal network, but IoT malware has exploited those who do not filter this at the firewall and expose it to the Internet
  - Typically inbound on the router itself as a path into the network and discovery
  - Mirai, etc.

EXPLOITATION OF SSDP ULA NTP FIELD - UDP COVERT

- Router with UPnP enabled could respond
- And Respond back with packet that contains dead drop URL or hidden message
- Internet probe M-SEARCH Query at Router
- Response includes OPT field

Covert UDP - SSDP

Smart Plug

Two-way conversations using SSDP to hide content (M2M)

- Smart Plug sends M-Search
- M-Search packet embeds hidden message or CnC URL in OPT filed
- Received by other IoT devices on network

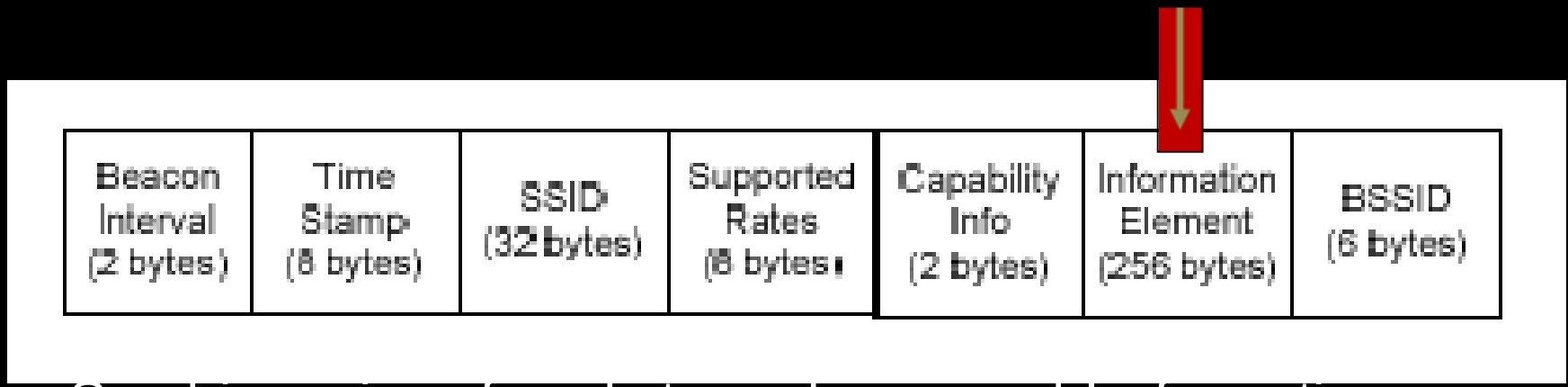
Smart TV

- Smart TV receives M-Search Packet and responds
- NOTIFY Packets send a packet back with embedded response

Covert 802.11 (WiFi Layer 2)

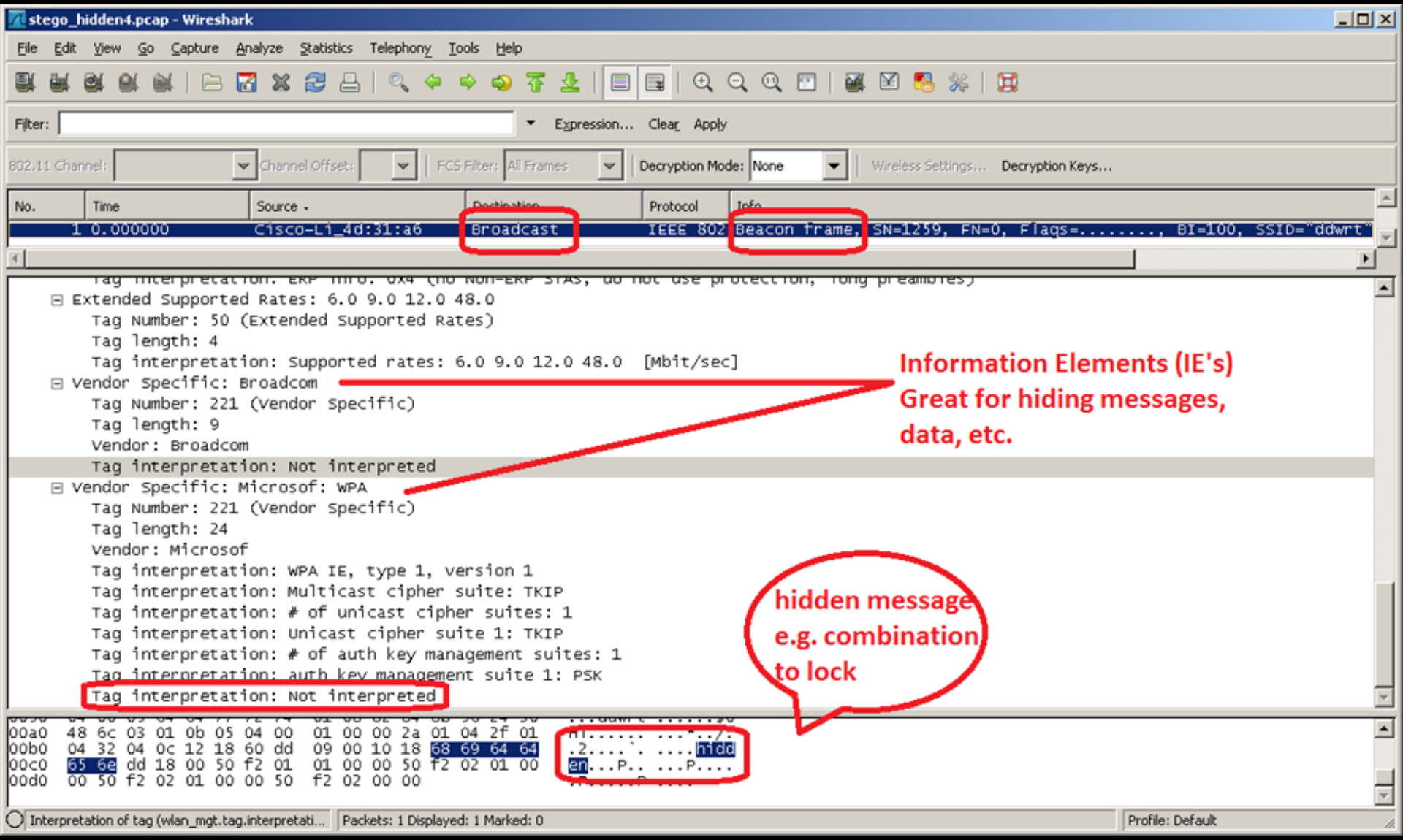
WiFi Beacon Stuffing

- Makes use of the Information Elements (IE) found in a Beacon Packet
- Can add up to 253 bytes of vendor-specific info



- Send in series of packets and reassemble for entire message, or good for small IM-like messages

WiFi Stego Stuffing



WiFi Stego Stuffing

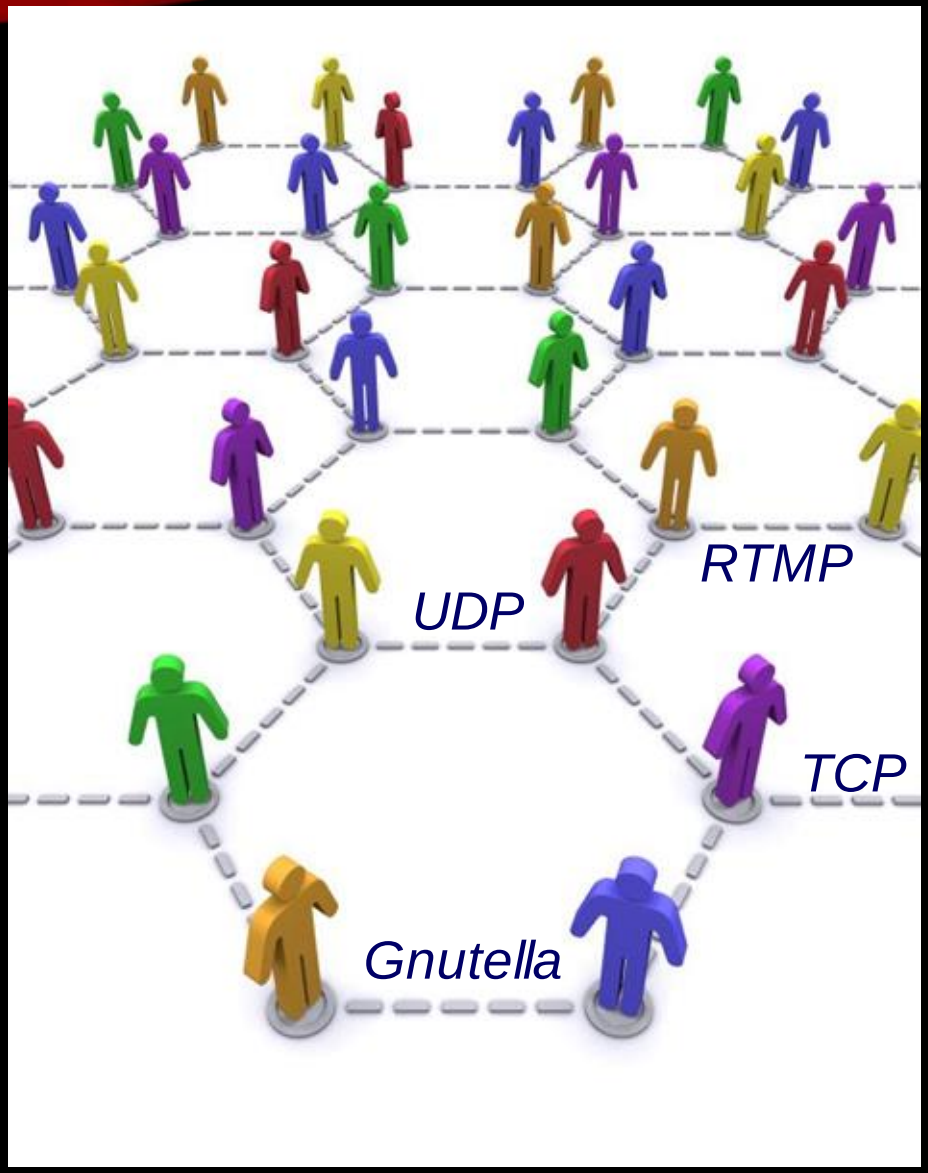
- Modify Beacon packet and insert message or data into IE field
- Then use aireplay-ng to replay pcap file to transmit to receiver on the other end

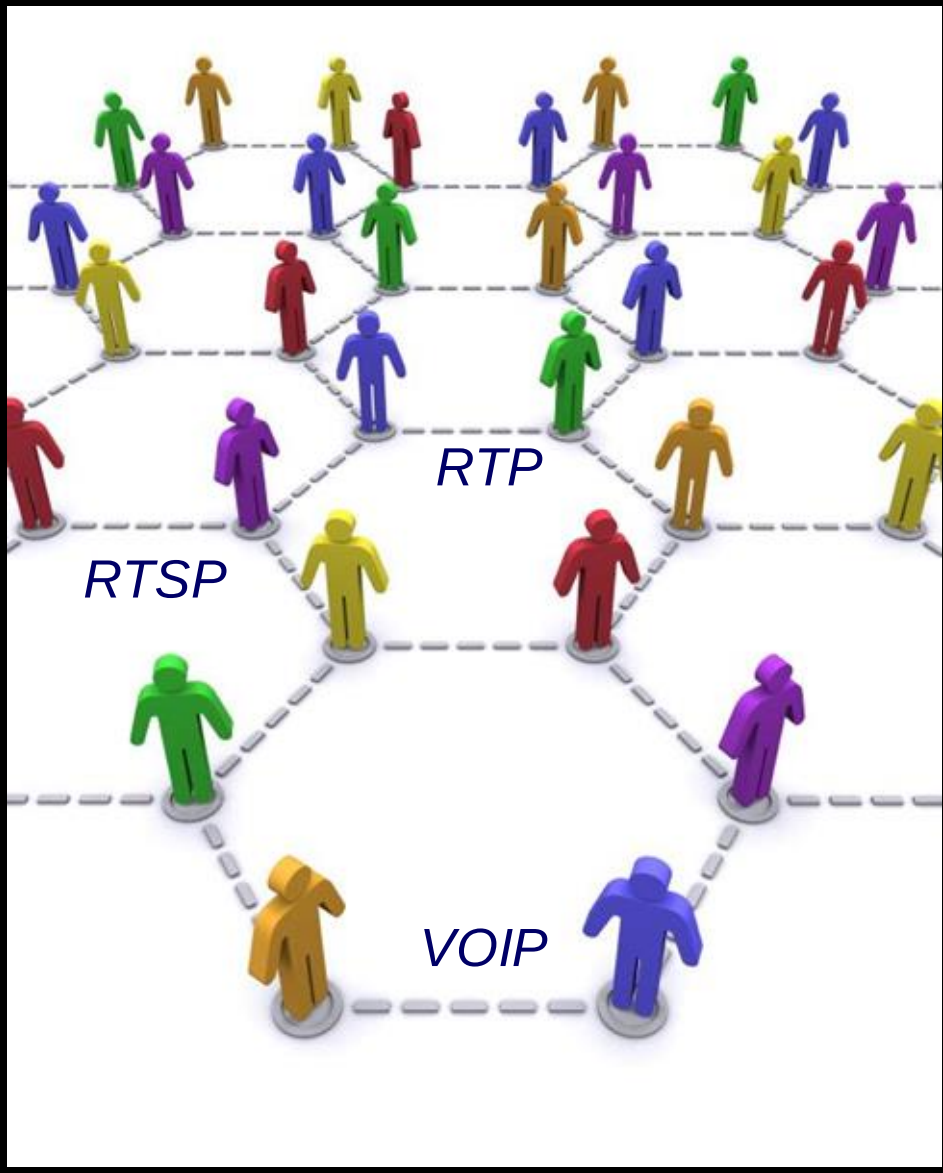
Receiver opens packet revealing message

- # aireplay-ng -r *.pcap -<interface>
- Prototyped on ddwrt

A MODERN TWIST STREAMING DATA

COVERT COMMUNICATION CHANNELS





STREAMING PACKETS

Streaming communications are ubiquitous today

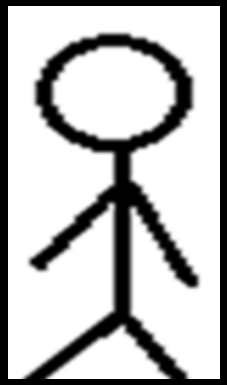
- The sheer volume of streaming packets flowing in and out of organizations today is staggering
- Other than encryption, much of this streaming traffic goes unchecked due to several factors
  - Complexity of the encoding methods
  - Large number of packets
  - Relatively small size of each packet
  - Greater focus today is on web and e-mail

STREAMING PACKETS

Alice

P1 P2 P3 P4 Pn

Pn P4 P3 P2 P1



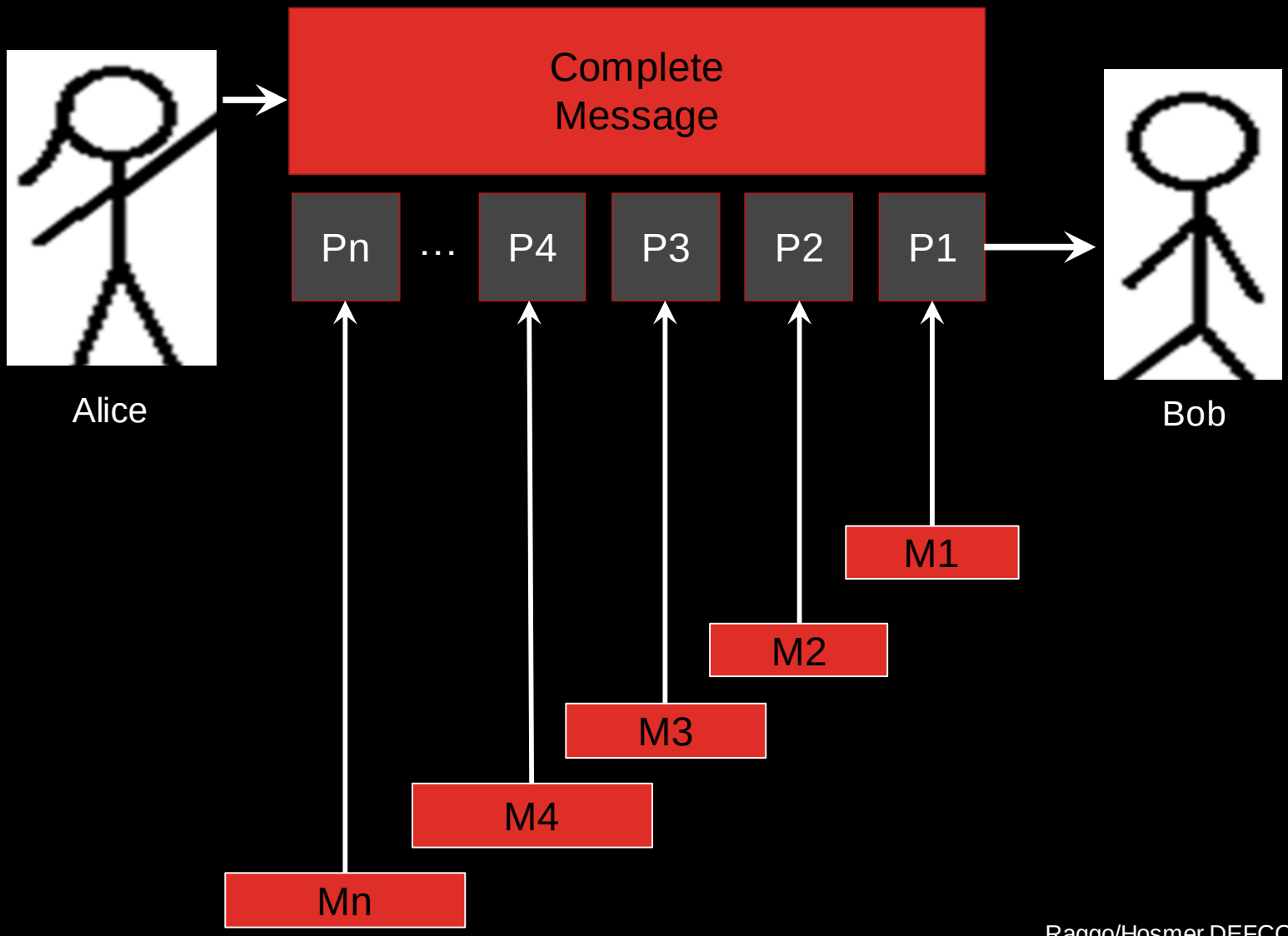
Bob

Header

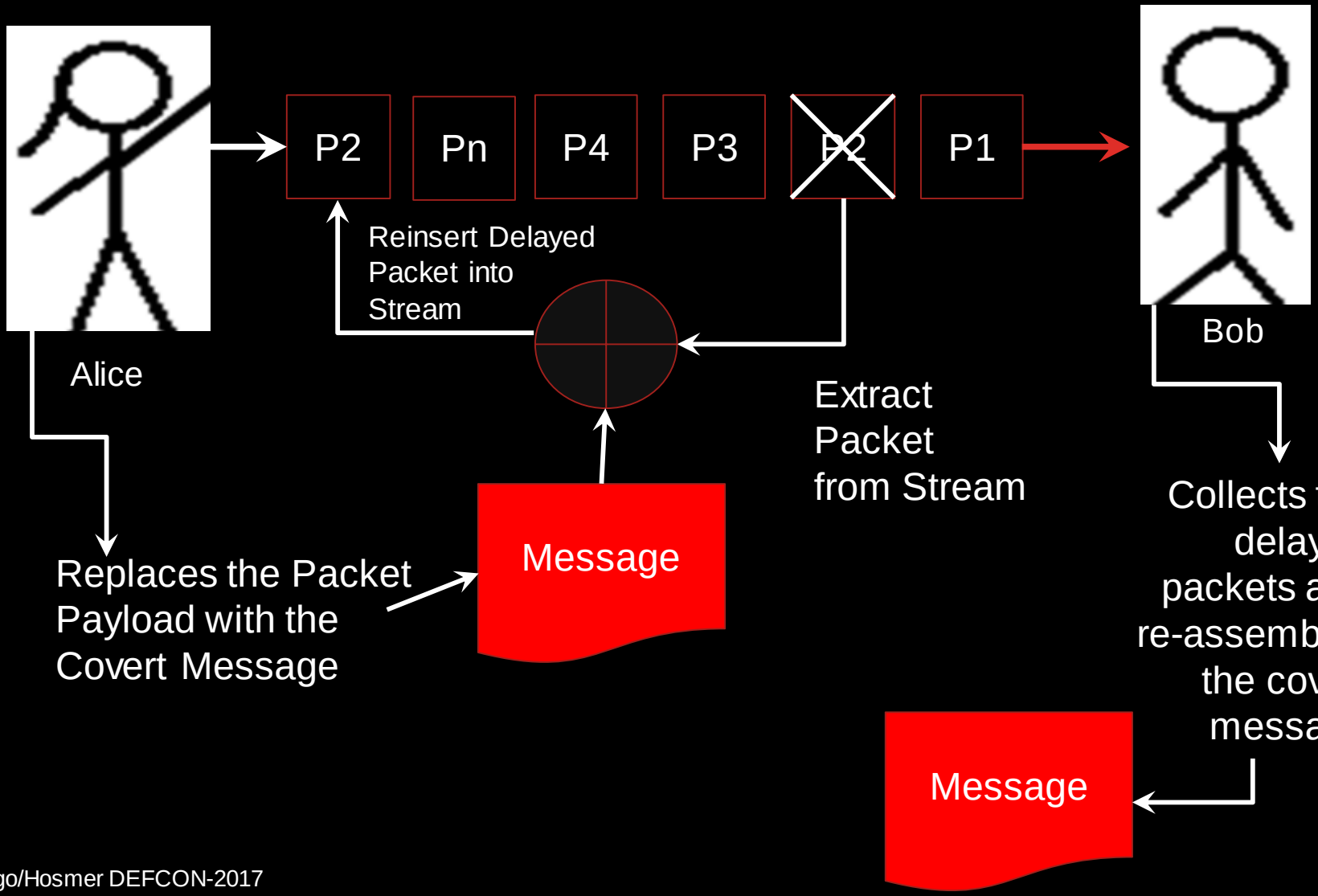
Payload

Numerous Payload CODECS types exist, each with differing structural elements

STREAMING PAYLOAD TRANSMISSION



DELAYED PACKET INSERTION



SUMMARY OF DELAYED PACKET INSERTION

- **Advantages**
  - Volume of streaming based traffic is enormous
  - Most if not all of the content of this traffic goes unchecked
- **Disadvantages**
  - More complex to pull off
  - Normal data loss can corrupt or lose message content