

For Fun and Profit

Build your stack with Scapy

Disclaimer

We do not pretend to know everything about the topics discussed in this workshop. If you happen to know that we are for a fact wrong about something, don't just sit there quietly, SPEAK UP.

If you have a question about anything discussed in this workshop, SPEAK UP. There are no stupid or dumb questions. Odds are, the smart looking person next to you is more lost than you are. SPEAK UP, and ask those questions!

We won't get mad at ya for doing so. We might even buy you a beer!

ping stryngs

- UAV operator USMC
- Linux since 2006
- Information Assurance – 4+ Years
- <3 802.11

ping Jack64

- Does AppSec stuff for Checkmarx (2016-)
 - Cobalt Core Researcher
- 3+ years experience in infosec

nmap workshop -vvv

- tmux/Screen
- iPython
- Primer on 802.11
- Primer on TCP
- Scapy
 - Auth/assoc
 - kSnarf
 - Out of Band communications
 - airpwn-ng
 - pyDot11
 - Utils (pcap-geolocate, port-knocking backdoor)
- Comments / Questions / Tomato throwing session

Primer Questions

- Q: How many of you know what a raw HTTP GET looks like?

Primer Questions

- Q: How many of you know what a raw HTTP GET looks like?
- Q: How many of you know how TCP/IP works?

Primer Questions

- Q: How many of you know what a raw HTTP GET looks like?
- Q: How many of you know how TCP/IP works?
- Q: How many of you have played with raw packets before?

Primer Questions

- Q: How many of you know what a raw HTTP GET looks like?
- Q: How many of you know how TCP/IP works?
- Q: How many of you have played with raw packets before?
- Q: How many of you are comfortable in python?

Primer Questions

- Q: How many of you know what a raw HTTP GET looks like?
- Q: How many of you know how TCP/IP works?
- Q: How many of you have played with raw packets before?
- Q: How many of you are comfortable in python?
- Q: How many of you are blue teamers?

Primer Questions

- Q: How many of you know what a raw HTTP GET looks like?
- Q: How many of you know how TCP/IP works?
- Q: How many of you have played with raw packets before?
- Q: How many of you are comfortable in python?
- Q: How many of you are blue teamers?
- Q: How many of you are red teamers?

tmux / Screen

- Used as a terminal multiplexer
- Allows you to re-ssh without nohup nonsense
- Essential for headless
- Saves real-estate
- Scapy wants shells.

IPython

- ncurses based
 - We don't need no stinkin GUI
- Tab it out
- ?
- When used with Scapy
 - Endless results
 - Severe happiness
 - Pwnage

802.11

- Radio waves are all around us
- Works off the concept of “Shared Medium”
- All we have to do, is listen
 - Can't hide from me....



Useful 802.11 Acronyms

- Authenticator: An entity at one end of a point-to-point LAN segment that facilitates authentication of the entity attached to the other end of that link. (IEEE Std 802.1X-2004)
- BSSID: Basic Service Set Identifier. Commonly known as the MAC address of a specific Access Point.
- Distribution system (DS): A system used to interconnect a set of basic service sets (BSSs) and integrated local area networks (LANs) to create an extended service set (ESS).
- ESSID: Extended Service Set Identifier. Commonly referred to as the SSID and known by the masses as the “Network Name”.

Useful 802.11 Acronyms

- Group Temporal Key (GTK): A random value, assigned by the group source, which is used to protect group addressed medium access control (MAC) protocol data units (MPDUs) from that source. The GTK might be derived from a group master key (GMK).
- Medium Access Control (MAC) Protocol Data Unit (MPDU): The unit of data exchanged between two peer MAC entities using the services of the physical layer (PHY).
- Pairwise Master Key (PMK): The key derived from a key generated by an Extensible Authentication Protocol (EAP) method or obtained directly from a preshared key (PSK).
- Pairwise Transient Key (PTK): A concatenation of session keys derived from the pairwise master key (PMK) or from the PMK-R1. Its components include a key confirmation key (KCK), a key encryption key (KEK), and one or more temporal keys that are used to protect information exchanged over the link.

Useful 802.11 Acronyms

- Station (STA): A logical entity that is a single addressable instance of a medium access control (MAC) and physical layer (PHY) interface to the wireless medium (WM).
- Supplicant: An entity at one end of a point-to-point LAN segment that is being authenticated by an Authenticator attached to the other end of that link. (IEEE Std 802.1X-2004)

From-DS and To-DS

Table 8-2—To/From DS combinations in data frames

To DS and From DS values	Meaning
To DS = 0 From DS = 0	A data frame direct from one STA to another STA within the same IBSS, a data frame direct from one non-AP STA to another non-AP STA within the same BSS, or a data frame outside the context of a BSS, as well as all management and control frames.
To DS = 1 From DS = 0	A data frame destined for the DS or being sent by a STA associated with an AP to the Port Access Entity in that AP.
To DS = 0 From DS = 1	A data frame exiting the DS or being sent by the Port Access Entity in an AP, or a group addressed Mesh Data frame with Mesh Control field present using the three-address MAC header format.
To DS = 1 From DS = 1	A data frame using the four-address MAC header format. This standard defines procedures for using this combination of field values only in a mesh BSS.

Address Fields

From DS Set, To DS Clear:

Address 1: Destination
Address 2: BSSID
Address 3: Source

From DS Clear, To DS Clear:

Address 1: Destination
Address 2: Source
Address 3: BSSID

From DS Clear, To DS Set:

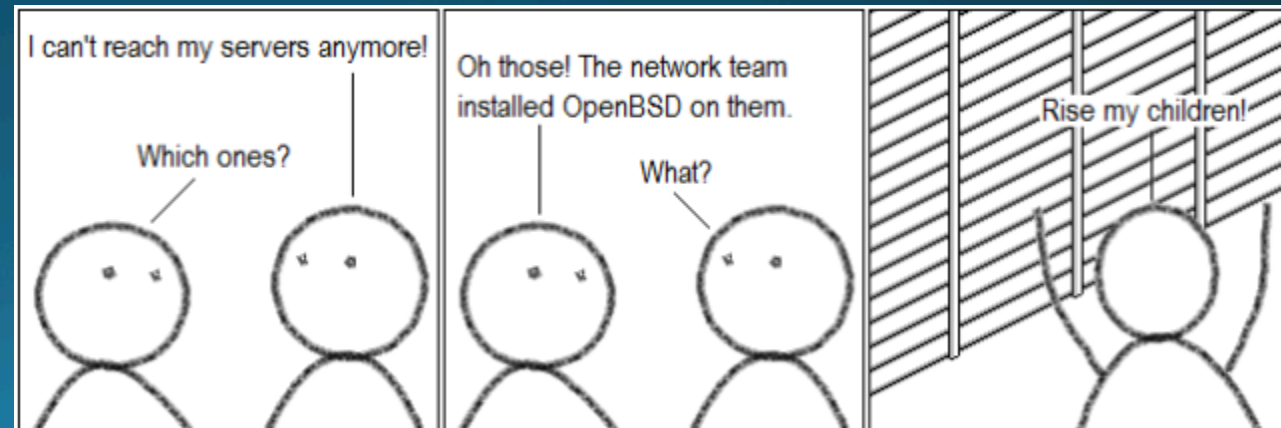
Address 1: BSSID
Address 2: Source
Address 3: Destination

From DS Set, To DS Set:

Address 1: Receiver
Address 2: Transmitter
Address 3: Destination
Address 4: Source

Crazy Memory Time

- Remember To First
 - 'Tis better TO give than to receive
- Next remember BSD
 - That crazy operating system TO give to your friends
 - BSSID
 - SOURCE
 - DESTINATION
 - To-DS



Crazy Memory Time

- Smart people use Databases (DBS), not Excel
 - Excel Sucks
 - Destination
 - BSSID
 - Source
 - From-DS



Crazy Memory Time

- Learn your Scapy Decimals for Direction
 - 1 - To-DS
 - 2 - From-DS
 - 65 - To-DS (with Encryption)
 - 66 - From-DS (with Encryption)

Crazy Memory Time

- Learn your Scapy Decimals for Type
 - 0 - Management
 - 1 - Control
 - 2 - Data

“First you have to Manage, then you can Control the Data”

Crazy Memory Time

- All of this corresponds to the Dot11 Class
 - FCfield
 - addr1
 - addr2
 - addr3
 - type
- Learn how to manipulate and interpret the above
 - The world is your oyster

TCP

- What do you think of when you hear TCP-IP?

TCP

- What do you think of when you hear TCP-IP?
 - syn , syn/ack, ack
 - FIN, PSH, URG

TCP

- What do you think of when you hear TCP-IP?
 - syn , syn/ack, ack
 - FIN, PSH, URG
- How do you attack TCP?

TCP

- What do you think of when you hear TCP-IP?
 - syn , syn/ack, ack
 - FIN, PSH, URG
- How do you attack TCP?
 - Sequence and Acknowledgement numbers
 - First horse across the finish line, wins



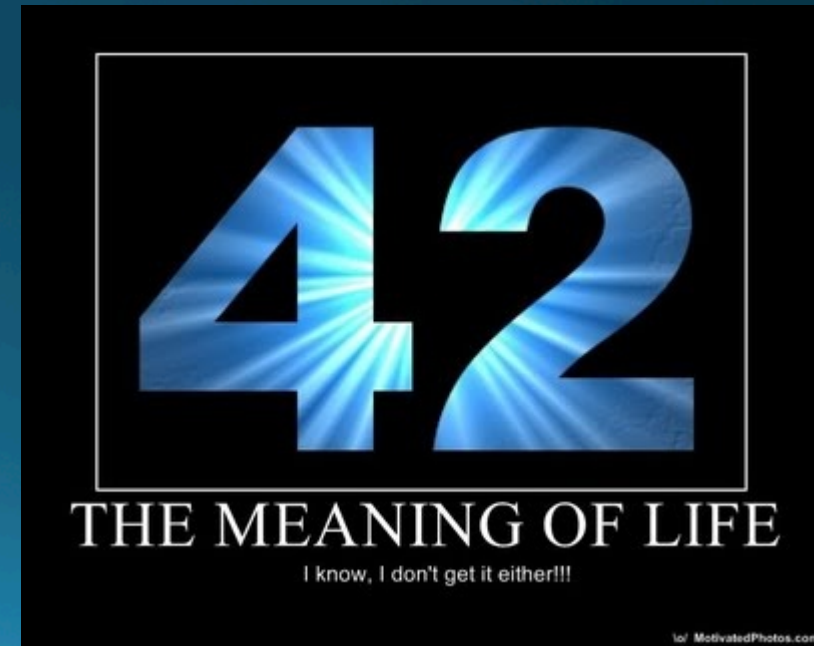
TCP

- What do you think of when you hear TCP-IP?
 - syn , syn/ack, ack
 - FIN, PSH, URG
- How do you attack TCP?
 - Sequence and Acknowledgement numbers
 - First horse across the finish line, wins
- If you remember nothing else



TCP

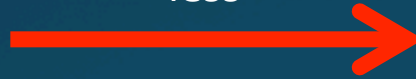
- What do you think of when you hear TCP-IP?
 - syn , syn/ack, ack
 - FIN, PSH, URG
- How do you attack TCP?
 - Sequence and Acknowledgement numbers
 - First horse across the finish line, wins
- If you remember nothing else
 - Sequence + Length == Acknowledgement
 - This is your 42



TCP Flags



less

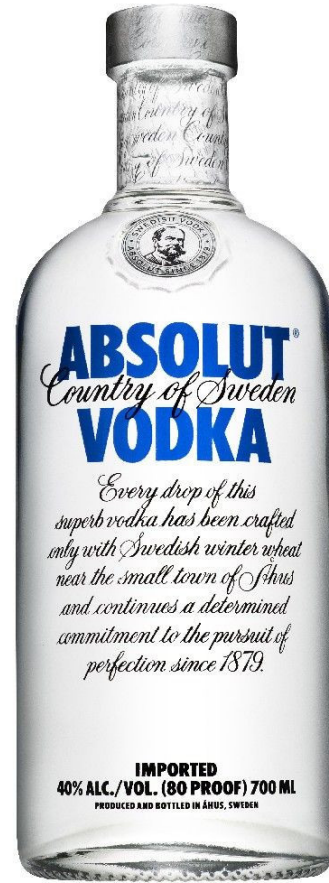
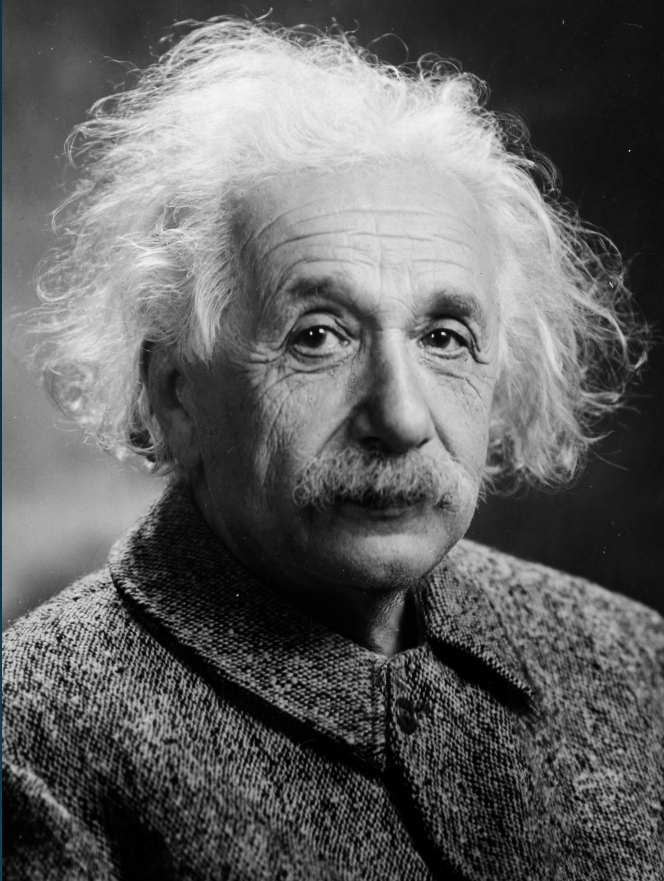


TCP Flags

- **FIN** – No more data from sender (Seen after a connection is closed)
- **URG** – indicates that the URgent pointer field is significant
- **CWR** – Congestion Window Reduced (CWR) flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set (added to header by [RFC 3168](#)).
- **ACK** – indicates that the ACKnowledgment field is significant (Sometimes abbreviated by tcpdump as ".")
- **SYN** – Synchronize sequence numbers (Seen on new connections)
- **PSH** – Push function
- **ECE** (ECN-Echo) – indicate that the TCP peer is ECN capable during 3-way handshake (added to header by [RFC 3168](#)).
- **RST** – Reset the connection (Seen on rejected connections)

TCP #s

-VS-



TCP #s

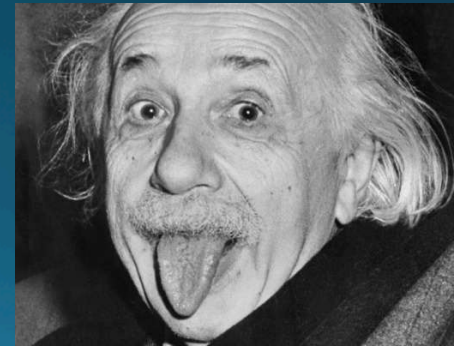
1 0.000000	192.168.1.125	192.168.1.132	TCP	74 37310 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=37983574 TSecr=0 WS=1024
2 0.000396	192.168.1.132	192.168.1.125	TCP	74 80 → 37310 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=359152 TSecr=37983574 WS=128
3 0.000452	192.168.1.125	192.168.1.132	TCP	66 37310 → 80 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=37983574 TSecr=359152
4 0.000544	192.168.1.125	192.168.1.132	HTTP	143 GET / HTTP/1.1
5 0.000790	192.168.1.132	192.168.1.125	TCP	66 80 → 37310 [ACK] Seq=1 Ack=78 Win=29056 Len=0 TSval=359152 TSecr=37983574
6 0.002040	192.168.1.132	192.168.1.125	HTTP	338 HTTP/1.1 200 OK (text/html)
7 0.002066	192.168.1.125	192.168.1.132	TCP	66 37310 → 80 [ACK] Seq=78 Ack=273 Win=30720 Len=0 TSval=37983574 TSecr=359152
8 0.002241	192.168.1.125	192.168.1.132	TCP	66 37310 → 80 [FIN, ACK] Seq=78 Ack=273 Win=30720 Len=0 TSval=37983574 TSecr=359152
9 0.004016	192.168.1.132	192.168.1.125	TCP	66 80 → 37310 [FIN, ACK] Seq=273 Ack=79 Win=29056 Len=0 TSval=359153 TSecr=37983574
10 0.004060	192.168.1.125	192.168.1.132	TCP	66 37310 → 80 [ACK] Seq=79 Ack=274 Win=30720 Len=0 TSval=37983575 TSecr=359153

-VS-

```
Out[9]: <Ether  dst=aa:bb:cc:dd:ee:ff src=08:00:27:0d:60:76 type=0x800 |<IP  version=4L ihl=5L tos=0x0 len=52 id=50023 flags=DF frag=0L ttl=64 proto=tcp chksum=0xf30a src=192.168.1.132 dst=192.168.1.125 options=[] |<TCP  sport=http dport=37310 seq=840022423 ack=454279889 dataofs=8L reserved=0L flags=A window=227 chksum=0x835d urgptr=0 options=[('NOP', None), ('NOP', None), ('Timestamp', (359152, 37983574))] |>>>
```

TCP #s

1 0.000000	192.168.1.125	192.168.1.132	TCP	74 37310 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=37983574 TSecr=0 WS=1024
2 0.000396	192.168.1.132	192.168.1.125	TCP	74 80 → 37310 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=359152 TSecr=37983574 WS=128
3 0.000452	192.168.1.125	192.168.1.132	TCP	66 37310 → 80 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=37983574 TSecr=359152
4 0.000544	192.168.1.125	192.168.1.132	HTTP	143 GET / HTTP/1.1
5 0.000790	192.168.1.132	192.168.1.125	TCP	66 80 → 37310 [ACK] Seq=1 Ack=78 Win=29056 Len=0 TSval=359152 TSecr=37983574
6 0.002040	192.168.1.132	192.168.1.125	HTTP	338 HTTP/1.1 200 OK (text/html)
7 0.002066	192.168.1.125	192.168.1.132	TCP	66 37310 → 80 [ACK] Seq=78 Ack=273 Win=30720 Len=0 TSval=37983574 TSecr=359152
8 0.002241	192.168.1.125	192.168.1.132	TCP	66 37310 → 80 [FIN, ACK] Seq=78 Ack=273 Win=30720 Len=0 TSval=37983574 TSecr=359152
9 0.004016	192.168.1.132	192.168.1.125	TCP	66 80 → 37310 [FIN, ACK] Seq=273 Ack=79 Win=29056 Len=0 TSval=359153 TSecr=37983574
10 0.004060	192.168.1.125	192.168.1.132	TCP	66 37310 → 80 [ACK] Seq=79 Ack=274 Win=30720 Len=0 TSval=37983575 TSecr=359153



Coooookies!

- HTTP is stateless
- Cookies create a bridge for this purpose
- Have the cookie == Access
 - Mostly...
- Demo



Scapy

- Created by Philippe Biondi
- Best damn interactive packet manipulator, ever.
- Who here has used?

Scapy tidbits

- Namespace overwrite warning...
 - ls()
- lsc()
- pkt.show()
- rdpcap()/wrpcap()
- PcapReader()/PcapWriter()
- hexstr
- str
- send()/sendp()
 - prn
 - Closure example
 - lfilter

Learning to fly

- sendp() and send() only “work” if the tgt gets a packet it expects
 - Garbage in, Garbage out
- So, how do we build?
 - We could learn by reading the RFC and building the packet from scratch...
 - OR:
 - We use a pre-existing packet/frame from a PCAP!
- Take the pre-existing packet, and reverse engineer
 - World domination follows shortly



This is my Rifle

- We now know how to listen
- We wait for the trigger, then pull



Out-of-Band Communications Channel

- Problem: How to exfiltrate data from:
 - A) Airgapped systems or networks (no internet connection)?
 - B) system with highly monitored or restricted internet access (DPI, egress filtering, etc)?

Out-of-Band Communications Channel

- Answer: Use custom 802.11 frames!

Out-of-Band Communications Channel

```
~/smuggler
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
[]

mon0 (victim)
```

```
~/smuggler
~/smuggler# python smuggler.py

Tom Neaves
tneaves@trustwave.com V1.0
Smuggler
The Wireless Shell

mon1 (attacker)

Smuggler# who
tom      tty7      2014-10-01 13:19
tom      pts/1      2014-10-01 13:46 (:0)
tom      pts/2      2014-10-01 13:48 (:0)
tom      pts/3      2014-10-01 14:59 (:0)
Smuggler# []
```

Out-of-Band Communications Channel

The image shows a Wireshark packet capture analysis of IEEE 802.11 Probe Requests. The filter is set to `wlan.fc.type_subtype == 0x04`. The packet list shows four probe requests from source MAC `c4:2f:xx:xx:xx:xx` to destination `ff:ff:ff:ff:ff:ff` on channel 124. The details pane for packet 629 shows the frame structure, including the IEEE 802.11 wireless LAN management frame and tagged parameters. The packet bytes pane shows the raw data, with the SSID 'tom' and timestamp '2014-10-01 13:19' highlighted in red.

No.	Time	Source	Destination	Protocol	Length	Info
629	20.51222300	c4:2f:xx:xx:xx:xx	ff:ff:ff:ff:ff:ff	802.11	124	Probe Request, SN=0, FN=0, Flags=.....C, SSID=tom, tty7, 2014-10-01 13:19
634	20.67576000	61:52:xx:xx:xx:xx	ff:ff:ff:ff:ff:ff	802.11	129	Probe Request, SN=0, FN=0, Flags=.....C, SSID=tom, pts/1, 2014-10-01 13:46 (:0)
635	20.83147100	d9:3f:xx:xx:xx:xx	ff:ff:ff:ff:ff:ff	802.11	129	Probe Request, SN=0, FN=0, Flags=.....C, SSID=tom, pts/2, 2014-10-01 13:48 (:0)
637	20.99119800	93:7e:xx:xx:xx:xx	ff:ff:ff:ff:ff:ff	802.11	129	Probe Request, SN=0, FN=0, Flags=.....C, SSID=tom, pts/3, 2014-10-01 14:59 (:0)

Frame 629: 124 bytes on wire (992 bits), 124 bytes captured (992 bits)

Radiotap Header v0, Length 25

IEEE 802.11 Probe Request, Flags:C

- Type/Subtype: Probe Request (0x04)
- Frame Control: 0x0040 (Normal)
- Duration: 0
- Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
- Source address: c4:2f:xx:xx:xx:xx
- BSS Id: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
- Fragment number: 0
- Sequence number: 0
- Frame check sequence: 0x569b80ec [correct]

IEEE 802.11 wireless LAN management frame

- Tagged parameters (71 bytes)

0000 00 00 19 00 6f 08 00 00 1d 90 e0 7a 00 00 00 000... ..Z....

0010 10 02 8a 09 80 04 c6 a4 00 40 00 00 00 ff ff ff@.....

0020 ff ff ff c4 2f xx xx xx xx ff ff ff ff ff ff ff/.\$F M.....

0030 00 00 26 74 6f 6d 20 20 20 20 20 74 74 79 37 ..&tom tty7

0040 20 20 20 20 20 20 20 20 20 32 30 31 34 2d 31 30 2014-10

0050 2d 30 31 20 31 33 3a 31 39 2a 01 04 2f 01 04 32 -01 13:1 9*../..2

0060 04 0c 12 18 60 01 08 82 84 8b 96 24 30 48 6c 03\$OHL..

0070 01 03 05 04 00 01 00 00 56 9b 80 ecV...

Out-of-Band Communications Channel

AUGUST 4-7, 2016

Issues:

- Deep packet inspection firewalls may prove of trouble here.
- Reordering the data at receiver end could be an issue, should sequencing is not taken care of before shoving in the data.
- No retrieval of lost frames so far.



Out-of-Band Communications Channel

- Writing a detector for Smuggler/Covert Cupid/Chura-Liya:
 - Beacon frames with SSID field larger than 32 bytes
 - Probe Requests with Rates field larger than 32 bytes
 - Regex search for common unix words (root, bash, etc)

Out-of-Band Communications Channel

- A different approach - Use 802.11 Data Packets

Out-of-Band Communications Channel



Out-of-Band Communications Channel

- Detecting OOBBackdoor
- Protocol limitations
- Work-in-Progress

Scapy Utils

Geolocate-pcap.py

Port-Knocking Backdoor

airpwn-ng

- Conceptualized from the original airpwn by toast
- POC in bash
 - Remastered in Python
- Why does it work?
 - 1st to the finish line of course...

airpwn-ng

- Open wifi simple concept demonstration
 - This is our fastest example as no encryption needed

pyDot11

- On-the-fly encryption and decryption of 802.11
 - WEP or WPA
 - CCMP only right now, no TKIP...
- Created because stryngs could not find any other tool like it
 - airventriloquist-ng...
- Learned, so much
 - Passion and devotion pays off.
- Eventually will be scrapped in favor of airventriloquist-ng as a .so for Python

airpwn-ng

- Encrypted wifi demo

Questions?

