



Presenting

P@cketR@quet:

An Auditory IDS/Network Auralizer

Introduction

- Killian Ditch – killianditch@gmail.com
 - @killianditch / github.com/KillianDitch
- Penetration Tester for Coalfire Systems, Inc.
- Toiled through 8 years of IT and software support: helpdesk, sysadmin, developer, fix-it-now-or-get-blamed, etc.
- Moved to an offensive security consulting role relatively recently.

Agenda


- ~~Introduction~~
- Briefing – What & Why?
- Current Solutions
- PR Introduction
- Code Overview?
- Q&A

What?

- Tool Debut
 - First shared tool/code
- Only a basic level of networking knowledge assumed
 - Create foundations via demonstrations
- Slides + Hands-on

Why?



A decorative graphic in the bottom-left corner of the slide, consisting of several overlapping, semi-transparent geometric shapes in shades of gray and black, forming a stylized representation of a laptop's corner.

Why?

- Cool idea

Why? (The Formal Version)

- Alternative means of network analysis
 - Wireshark & tcpdump
- Experience network traffic in a new way – via sound
- Expanded appeal?
- Accessibility

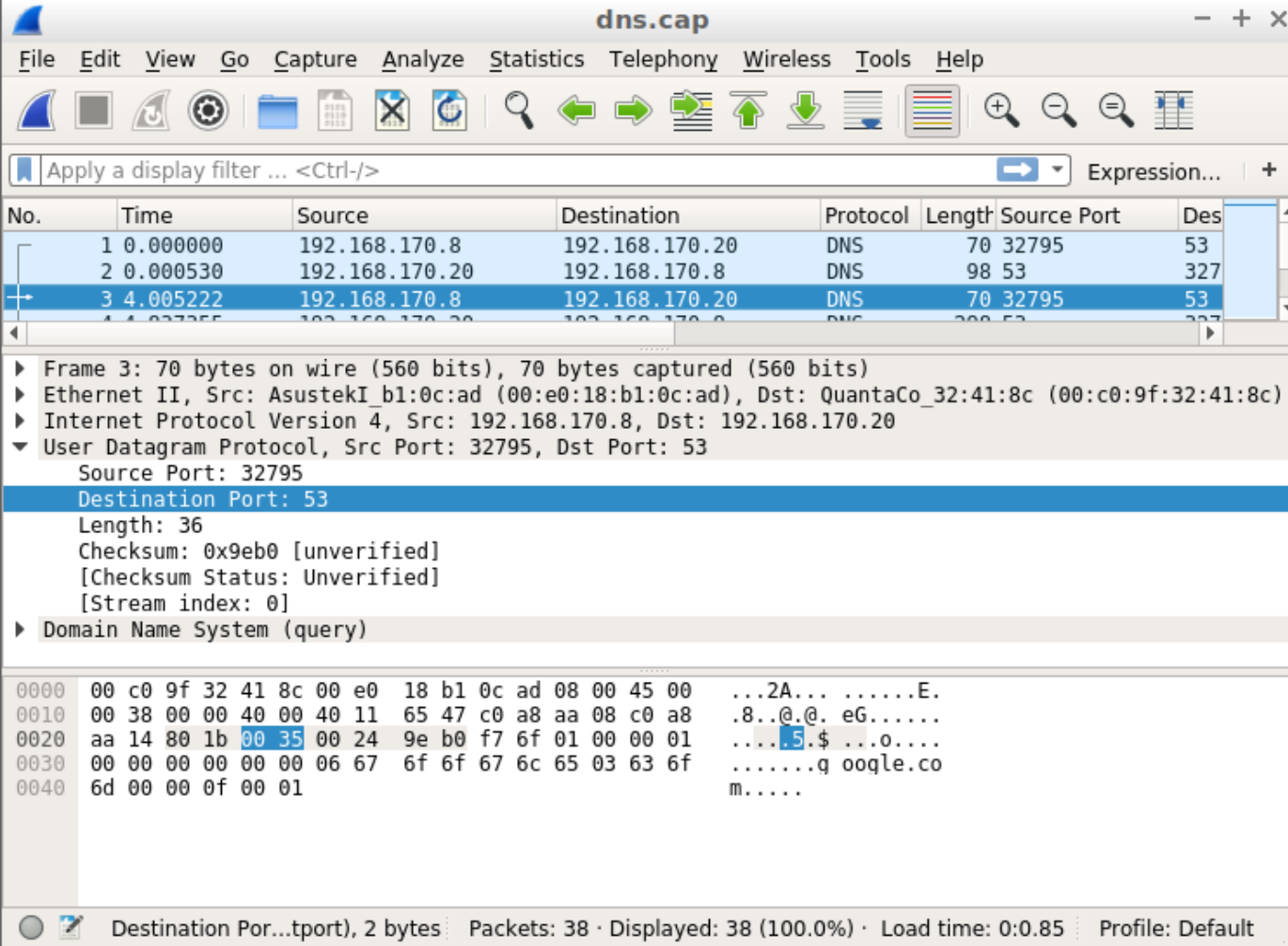
Standard Solutions – tcpdump

```
root@lUbuntu1:/home/kditch/pcaps# tcpdump -r dns.cap
```

```
reading from file dns.cap, link-type EN10MB (Ethernet)
```

```
02:47:46.496046 IP 192.168.170.8.32795 > 192.168.170.20.domain: 4146+ TXT? google.com. (28)
02:47:46.496576 IP 192.168.170.20.domain > 192.168.170.8.32795: 4146 1/0/0 TXT "v=spf1 ptr ?all" (56)
02:47:50.501268 IP 192.168.170.8.32795 > 192.168.170.20.domain: 63343+ MX? google.com. (28)
02:47:51.333401 IP 192.168.170.20.domain > 192.168.170.8.32795: 63343 6/0/6 MX smtp4.google.com. 40, MX smtp5.google.com. 10, MX smtp6.google.com. 10, MX smtp1.google.com. 10, MX smtp2.google.com. 10, MX smtp3.google.com. 40 (256)
02:47:59.313231 IP 192.168.170.8.32795 > 192.168.170.20.domain: 18849+ LOC? google.com. (28)
02:47:59.452255 IP 192.168.170.20.domain > 192.168.170.8.32795: 18849 0/0/0 (28)
02:48:07.320873 IP 192.168.170.8.32795 > 192.168.170.20.domain: 39867+ PTR? 104.9.192.66.in-addr.arpa. (43)
02:48:07.321379 IP 192.168.170.20.domain > 192.168.170.8.32795: 39867 1/0/0 PTR 66-192-9-104.gen.twtelecom.net. (87)
02:49:18.685951 IP 192.168.170.8.32795 > 192.168.170.20.domain: 30144+ A? www.netbsd.org. (32)
02:49:18.734862 IP 192.168.170.20.domain > 192.168.170.8.32795: 30144 1/0/0 A 204.152.190.12 (48)
02:49:35.461181 IP 192.168.170.8.32795 > 192.168.170.20.domain: 61652+ AAAA? www.netbsd.org. (32)
02:49:35.698849 IP 192.168.170.20.domain > 192.168.170.8.32795: 61652 1/0/0 AAAA 2001:4f8:4:7:2e0:81ff:fe52:9a6b (60)
02:50:35.523440 IP 192.168.170.8.32795 > 192.168.170.20.domain: 32569+ AAAA? www.netbsd.org. (32)
02:50:35.523827 IP 192.168.170.20.domain > 192.168.170.8.32795: 32569 1/0/0 AAAA 2001:4f8:4:7:2e0:81ff:fe52:9a6b (60)
02:50:44.735890 IP 192.168.170.8.32795 > 192.168.170.20.domain: 36275+ AAAA? www.google.com. (32)
02:50:44.752428 IP 192.168.170.20.domain > 192.168.170.8.32795: 36275 1/0/0 CNAME www.l.google.com. (52)
02:50:54.349862 IP 192.168.170.8.32795 > 192.168.170.20.domain: 56482+ AAAA? www.l.google.com. (34)
02:50:54.366527 IP 192.168.170.20.domain > 192.168.170.8.32795: 56482 0/0/0 (34)
02:51:35.204348 IP 192.168.170.8.32795 > 192.168.170.20.domain: 48159+ AAAA? www.example.com. (33)
02:51:35.437491 IP 192.168.170.20.domain > 192.168.170.8.32795: 48159 0/0/0 (33)
```


Standard Solutions – Wireshark



The image shows the Wireshark network protocol analyzer interface. The title bar indicates the file is 'dns.cap'. The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains various icons for file operations, capture control, and analysis. The display filter bar shows 'Apply a display filter ... <Ctrl-/>' and 'Expression...'. The packet list pane shows four packets, with packet 3 selected. The packet details pane shows the structure of packet 3: Frame 3 (70 bytes on wire, 70 bytes captured), Ethernet II (Src: AsustekI_b1:0c:ad, Dst: QuantaCo_32:41:8c), Internet Protocol Version 4 (Src: 192.168.170.8, Dst: 192.168.170.20), and User Datagram Protocol (Src Port: 32795, Dst Port: 53). The packet bytes pane shows the raw data in hexadecimal and ASCII, with the ASCII representation being '...2A... ..E. .8..@.@. eG.....5.\$...o....q ooqle.co m.....'. The status bar at the bottom shows 'Destination Port... (port), 2 bytes', 'Packets: 38 · Displayed: 38 (100.0%)', 'Load time: 0:0.85', and 'Profile: Default'.

No.	Time	Source	Destination	Protocol	Length	Source Port	Destination Port
1	0.000000	192.168.170.8	192.168.170.20	DNS	70	32795	53
2	0.000530	192.168.170.20	192.168.170.8	DNS	98	53	32795
3	4.005222	192.168.170.8	192.168.170.20	DNS	70	32795	53
4	4.007355	192.168.170.20	192.168.170.8	DNS	98	53	32795

Frame 3: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface
Ethernet II, Src: AsustekI_b1:0c:ad (00:e0:18:b1:0c:ad), Dst: QuantaCo_32:41:8c (00:c0:9f:32:41:8c)
Internet Protocol Version 4, Src: 192.168.170.8, Dst: 192.168.170.20
User Datagram Protocol, Src Port: 32795, Dst Port: 53
Source Port: 32795
Destination Port: 53
Length: 36
Checksum: 0x9eb0 [unverified]
[Checksum Status: Unverified]
[Stream index: 0]
Domain Name System (query)

0000 00 c0 9f 32 41 8c 00 e0 18 b1 0c ad 08 00 45 00 ...2A... ..E.
0010 00 38 00 00 40 00 40 11 65 47 c0 a8 aa 08 c0 a8 .8..@.@. eG.....
0020 aa 14 80 1b 00 35 00 24 9e b0 f7 6f 01 00 00 015.\$...o....
0030 00 00 00 00 00 00 06 67 6f 6f 67 6c 65 03 63 6fq ooqle.co
0040 6d 00 00 0f 00 01 m.....

Destination Port... (port), 2 bytes | Packets: 38 · Displayed: 38 (100.0%) | Load time: 0:0.85 | Profile: Default

P@cketR@quet

- TLDR: Nifty Python tool to play music corresponding to network traffic that contains the potential for an accessibility function for the visually-impaired.
- Name: the tool redirects packets from their digital form to audio; a tennis racket for packets.

Theory

- Provide insight into normal and aberrant traffic patterns by creating a sound-based representation of network traffic.
- Establish a baseline sound profile for normal traffic, thereby causing oddities such as ICMP, ARP, or UDP/TCP port scans to stand out.

Purpose

- Anyone interested in keeping track of the network, whether an analyst or tester, can listen to the sounds of the packets instead of scrolling through Wireshark or tcpdump output.
- Monitor service responses.
- Aberrant floods.

Accessibility

- Visually-impaired individuals could be trained in the notes and corresponding packets and then be empowered to conduct hitherto inaccessible network analysis.

Example

- If a port scan was observed by the monitoring interface, those packets would correspond to different sounds, thereby yielding an aural experience matching that traffic pattern.

Prove It!

- basic_r@quet.py
 - Basic, all-in-one script with standard dependencies only
 - Currently missing pcap reading feature
- p@cketr@quet.py
 - Scapy!
 - Sound options

Known Issues/Future Plans

- Rewrite in Python3
- Implement pcap functionality in basic_r@quet
- Expand protocol coverage
- Tune catalog

Known Issues/Future Plans

- Attacking/Defending flag
- Performance/Scaling

Recap

- P@quetR@quet will not do away with Wireshark and tcpdump or their like.
- It may serve as a companion tool.
- Most importantly, it's fun.

Questions/Comments/Discussion?

