A-11: Trace File Anonymization

Thursday June 16, 2011. 1:30pm – 3:00pm

Jasper Bongertz

Senior Consultant | Fast Lane Institute for Knowledge Transfer

SHARKFEST **'11**
Stanford University
June 13-16, 2011

Agenda

- Motivation
- Techniques
- Challenges
- Tools

Motivation for this talk

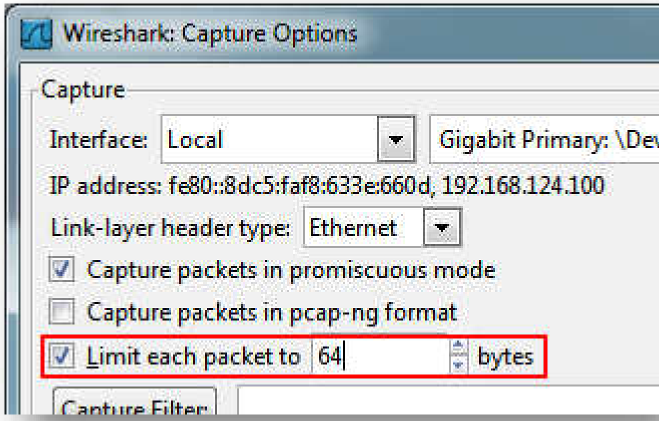
- First of all, I need to anonymize traces myself
  - Training material for student
  - Sharkfest talks („Have Wireshark will travel“)
- Question asked on ask.wireshark.org
  - http://ask.wireshark.org/questions/844/utility-to-anonymize-capture-files
- Apologies
  - I focussed on just a few tools. If you wrote one and it isn‘t included: sorry! Mail me. Maybe I do an updated talk sometime, somewhere ☺

Why trace file anonymization?

- Removing sensitive information from trace files
- Personal privacy
  - User IDs, passwords, IP addresses,...
- Confidential corporate information
  - Network topology
  - Potential choke points (DoS/DDoS)
  - Device & Software version information (CDP, LLDP)
  - Vulnerable protocols

Ways to anonymize

- Even Wireshark can do it!
  - Capture data with „Limit each packet to...“
  - Example: SMTP traffic patterns
- Can also be done after capture using „`editcap -s <snaplen>`“
- Using capture filters to exclude sensitive packets
  - filter on VLAN tags, Ethernet or IP addresses, TCP/UDP ports



Wireshark: Capture Options

Capture

Interface: Local | Gigabit Primary: \De

IP address: fe80::8dc5:faf8:633e:660d, 192.168.124.100

Link-layer header type: Ethernet

[✓] Capture packets in promiscuous mode

[ ] Capture packets in pcap-ng format

[✓] Limit each packet to 64 bytes

Capture Filter:

Ways to anonymize

- On-the-fly anonymization
  - Anonymize tracefiles while they‘re captured
  - Modification is done between capturing the packets and writing them to disk
- Challenges
  - Anonymization process must be fast enough to avoid drops
  - Original data can‘t be examined as it is never written to disk

Trace file modification

- Replace sensitive information with harmless information
- „Seek and destroy“
  - Look for parts that need to be replaced and replace them
  - Might overlook details in „unexpected places“
  - Problem with unknown protocols

Trace file modification

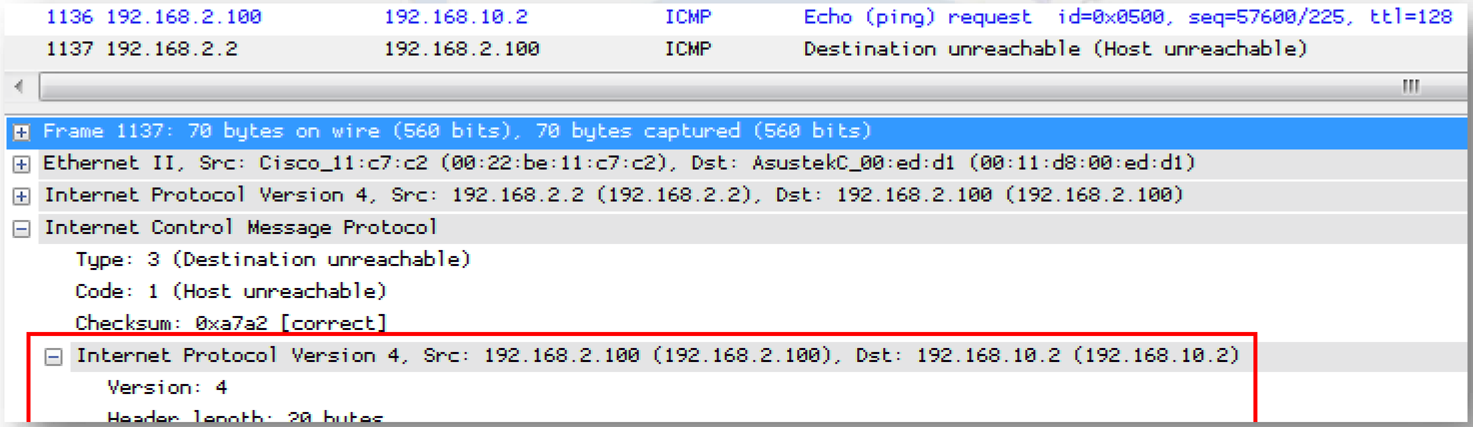
- „Defensive Transformation“
  - Copies and anonymizes known protocol header values
  - Removes or wipes all unknown parts
  - No chance of keeping sensitive information by mistake
- Core problem: layer 5-7 content
  - No well known header formats
  - Unknown UDP/TCP payload data format (if formatted at all)

Layers 1-4 – interesting data

- Ethernet
  - MAC vendor code (products used in the company)
- IP
  - Source, Destination address (network ranges)
  - Time To Live (topology)
- TCP
  - Source, Destination Port (Protocols in use)
  - Sequence Numbers (predictable?)
- UDP
  - Source, Destination Port (Protocols in use)

Layers 1-4 – interesting data

- ICMP
  - Destination Unreachable
  - Communication Filtered (ACL/Firewall)
  - TTL exceeded (Topology)
  - Redirect (Topology)
  - Quote of problem packet



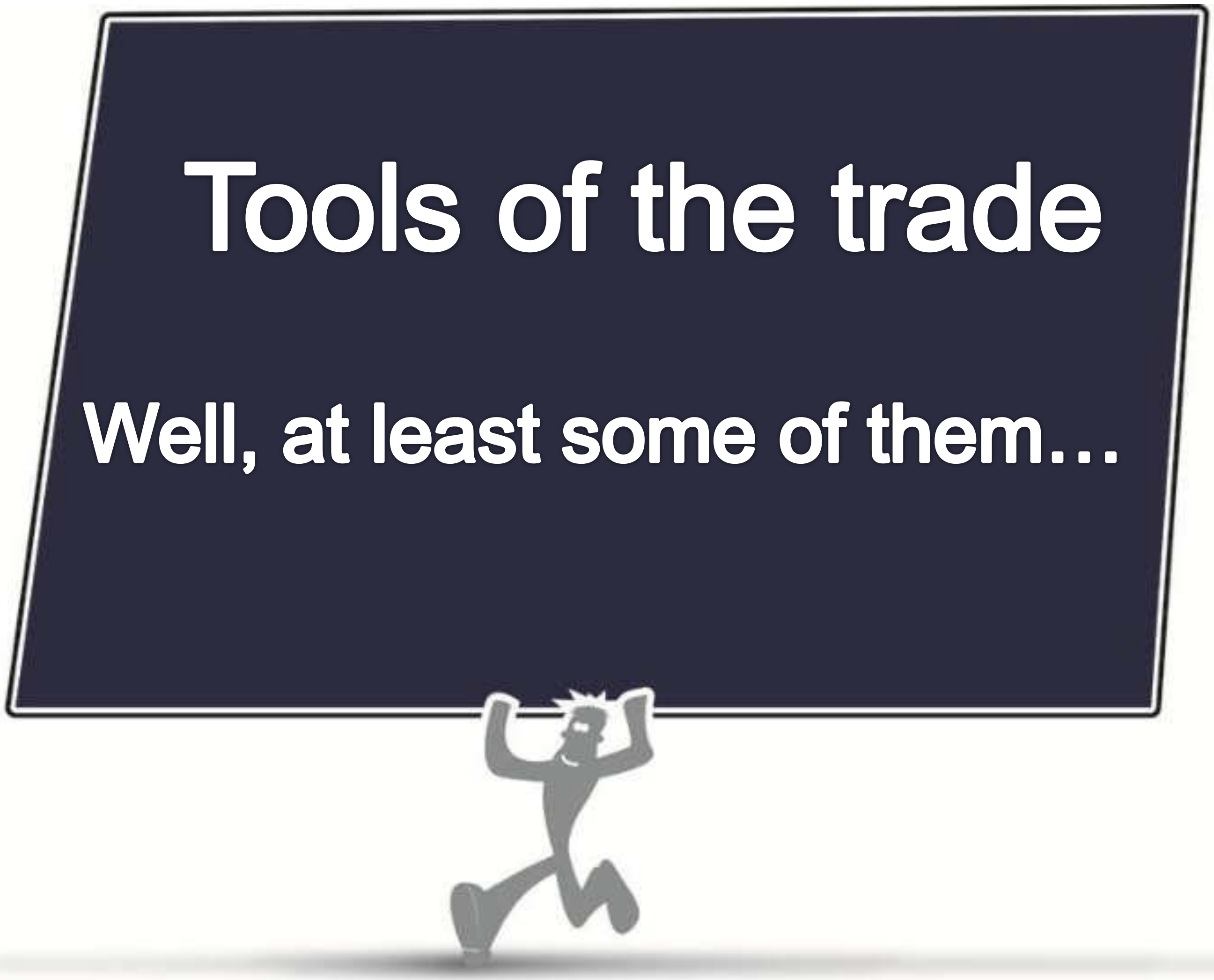
```
1136 192.168.2.100       192.168.10.2       ICMP      Echo (ping) request  id=0x0500, seq=57600/225, ttl=128
1137 192.168.2.2         192.168.2.100      ICMP      Destination unreachable (Host unreachable)

⊞ Frame 1137: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
⊞ Ethernet II, Src: Cisco_11:c7:c2 (00:22:be:11:c7:c2), Dst: AsustekC_00:ed:d1 (00:11:d8:00:ed:d1)
⊞ Internet Protocol Version 4, Src: 192.168.2.2 (192.168.2.2), Dst: 192.168.2.100 (192.168.2.100)
⊟ Internet Control Message Protocol
     Type: 3 (Destination unreachable)
     Code: 1 (Host unreachable)
     Checksum: 0xa7a2 [correct]
   ⊟ Internet Protocol Version 4, Src: 192.168.2.100 (192.168.2.100), Dst: 192.168.10.2 (192.168.10.2)
        Version: 4
        Header length: 20 bytes
```

Anonymization Challenges

- Keeping replacements consistent
  - Not only within one trace file, but across multiple trace files, too
  - MAC and IP addresses, TCP and UDP Ports
- Address range challenges
  - Replace IPs from the same subnet to end up in the same anonymized subnet
  - Class A/B/C easy compared to stuff like /29
  - Avoid randomizing broadcast addresses to unicast addresses

Address Replacement Strategies

- By specifying command line parameters
- Random numbers
  - Brute force method, hard to read a trace afterwards
- Using tables to keep replacements persistent
  - Might need lots of memory
  - Traces need to be processed sequentially
- Hashing original bytes to get a replacement



Tools of the trade

Well, at least some of them…

Tools not considered

- Some tools were not considered because we focus on reusability with Wireshark
  – We're here for packet analysis, so we like pcaps
  – Most users use Linux or Windows as a platform
- Sanitize, CryptoPAn
  – Output is ASCII, not a capture file
- tcpdpriv
  – Didn't work on Linux or Windows (probably my own fault though ☺)

The hard way

- Editing tracefiles manually, byte by byte
  - „when I was your age we did it like this:“

    ```
    echo „ÔÃ²¡                 ÿÿ     " > mycap.pcap
    ```

- Editing tracefiles using hex editors
  - Mostly for fixing checksum errors caused by tcp checksum offloading
  - Very slow
  - Easy to miss critical data when doing tracefile sanitation

Anonymization Tools

- Command line parameter tools
  - Editcap (just for truncating files)
  - Tcprewrite
  - Bittwiste
- „Fully automated“ tools
  - pktanon

Anonymization Tools - tcprewrite

- Command line tool
- Reads input file, replaces content specified by parameters, writes output file
- Can add/remove VLAN tags
- Can skip replacing broadcast addresses
- Replaces IP addresses consistently by using seeds
  - Also works with IPv6 ☺
- Subnet IP replacing with --pnat option

Anonymization Tools - tcprewrite

- Calculates correct checksums for edited frames
  - Information about existing bad checksums in source file is lost
- Allows recalculating bad checksums on purpose
  - Used to fix „checksum offloading“ problems
  - Important if trace has to be replayed

Anonymization Tools - tcprewrite

- --fixlen option to handle sliced frames
- Mostly needed for replay, not for packet analysis
- Three options:
  - *--fixlen=trunc* reduces the frame length to the number of captured bytes
  - *--fixlen=pad* adds 0x00 bytes to fill frame to match the original frame length
  - *--fixlen=del* deletes frames with original length bigger than captured size

Anonymization Tools - bittwiste

- Editor coming with bittwist (packet replay)
- Does not like some not so uncommon things
  - VLAN tags, IPv6 packets
- Lots of options similar to editcap
  - Saving by timeframe, packet range
- Allows deleting bytes from offsets
- -L option to truncate packets after layer 2-4
  - Unlike editcap, headers with options are not a problem

Anonymization Tools - bittwiste

- -C option to skip correcting checksums
  - crc of edited packets is always kept from the original trace -> does not match anymore
  - Without option set, crc is always recalculated, even for bad original frames
- -T option to modify headers
  - Need to specify layer: eth,arp,ip,tcp,udp,icmp
  - Must be last parameter
  - Only one –T option per run

Anonymization Tools - bittwiste

- IP address replacement options
  - Set source or destination to a specific IP
  - Replace existing source/destination IP with specific source/destination IP
  - No random/automatic IP replacing

Anonymization Tools - pktanon

- Uses XML files to specify
  - source/destination filename
  - anonymization options
- VLAN tags are not recognized
  - Trace is anonymized but crippled after Ethernet headers
- Defensive transformation
  - Only transfers and optionally anonymizes parts it understands
- Can handle encapsulations like IP-in-ICMP

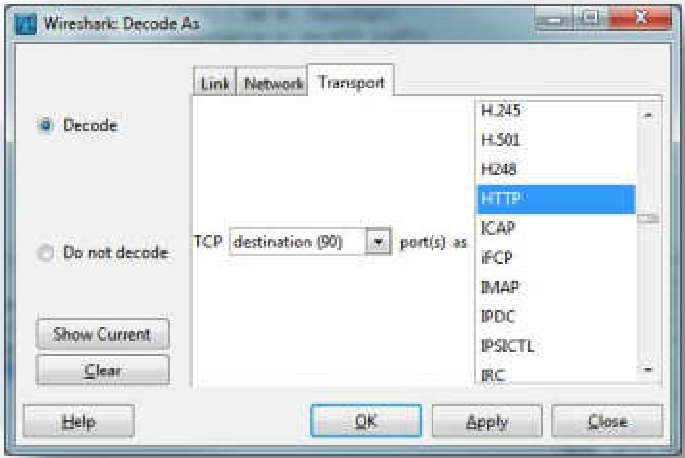
Anonymization Tools - pktanon

- Checksum recalculation option
  - Can keep bad original checksums with „SetBadChecksumsToBad“ option
- Anonymizing IP addresses
  - Achieving constistant IP address replacement using hashs of single octets
  - Keeps IPs from the same subnet in the same replacement subnet
  - Also works with IPv6 addresses

- Can handle layer 5-7 payloads
  - Replace with 0x00 or truncate payload
- Is able to anonymize tracefiles „on the fly“
  - `tcpdump -i eth0 -s 0 -w - | pktanon ./settings.xml`

Thinking out of the box

- Problem: sometimes, traces contain known protocols on weird port numbers
- Usually, we do „Decode As“ in Wireshark
- Setting is volatile between opening traces
- Trick: use tcprewrite or bittwiste to rewrite port numbers to a well known port
  - But keep the original as well, just to be sure!



TCP destination (90) port(s) as HTTP

H.245
H.501
H248
HTTP
ICAP
iFCP
IMAP
IPDC
IPSICTL
IRC

The anontracer VM

- VMware workstation machine
- tshark, tcpdump, editcap, tcprewrite, bittwiste and pktanon already installed and good to go
- Automatic mode for pktanon:
  - Drop a file into low/med/high directory
  - Runs anonymization according to .settings.xml
  - Anonymized file is put into same directory
  - Accessable via WebDav, ftp, SMB, ssh

Thanks

- I have to thank a couple of guys:
- Oliver Ripka, for creating the anontracer VM for me
- Eduard Blenkers, Christian Landström and Silvia Hagen, for providing fun traces
- The creators of tcprewrite, bittwiste and pktanon
  - Great job guys, I learned a lot while playing around with those

Questions?