Flow-level analysis: wireshark and Bro

Prof. Anja Feldmann, Ph.D.
Dr. Nikolaos Chatzis
wireshark – tshark

- Network packet analyzer for Unix/Windows
  - Displays detailed packet stats
  - GUI (wireshark) or command-line (tshark)

- Intended audience:
  - Network admins (troubleshooting)
  - Security engineers (security problems)
  - Developers (debugging protocols)
  - YOU 😊

- What it is not:
  - Not for large packet traces:
    - Not for high-speed links
    - Out of memory => crash!
tshark

Usage: tshark [options] ...

Capture interface:
   -i <interface>               name or idx of interface (def: first non-loopback)
   -f <capture filter>          packet filter in libpcap filter syntax
   -s <snaplen>                 packet snapshot length (def: 65535)
   -p                           don't capture in promiscuous mode
   -D                           print list of interfaces and exit
   -L                           print list of link-layer types of iface and exit
   -r <infile>                  set the filename to read from (no pipes or stdin!)

Processing:
   -R <read filter>             packet filter in Wireshark display filter syntax
   -n                           disable all name resolutions (def: all enabled)
   -d <layer_type>=<selector>,<decode_as_protocol> ...
   "Decode As", see the man page for details
   Example: tcp.port==8888,http
   -z <statistics>              various statistics, see the man page for details

Miscellaneous:
   -h                           display this help and exit
   -v                           display version info and exit
Basic stats with tshark

❖ Protocol summary of the trace:
  > tshark -q -z io,phs -r trace-1.pcap

❖ All traffic from/to a host every minute:
  > tshark -q -z io,stat,60,ip.addr==xxx -r trace-1.pcap

❖ All TCP conversations of the trace:
  > tshark -q -z conv/tcp -r trace-1.pcap

❖ All Telnet conversations of the trace:
  > tshark -q -z conv/tcp,telnet -r trace-1.pcap

❖ All UDP conversations of the trace:
  > tshark -q -z conv/udp -r trace-1.pcap

❖ All ICMP conversations of the trace:
  > tshark -q -z conv/tcp -r trace-1.pcap -R 'icmp'
Basic stats with wireshark

- General summary of the trace
- Protocol hierarchy stats
  - IP-level protocols
  - Transport protocols
  - ARP
  - ICMP
- "Conversations"
  - Follow a telnet session
  - Follow a DNS flow
  - Check IGMP messages
- Endpoints
  - Heavy-hitters
  - Low-hitters (scans)
- Packet size distribution
The Bro system

- Real-time network analysis framework
  - Unix-based network intrusion detection system
  - Misused for traffic analysis, e.g., by INET
- Emphasis on
  - Application-level semantics
    - Manipulating packets is uncommon/painful, e.g., wireshark
  - Tracking information over time
    - Within and across flows
    - Archiving for post-mortem analysis
- Scalability, i.e. Gbit/second links
The Bro system (2)

- Analyzing data means **programming** the analysis
  - No specification
  - No magic in Bro: The user has to specify what has to be detected

- Programming the analysis ~ behavioral analysis
  - No good/evil
  - But matched/unmatched
Connection summaries

- One line summary for all connections
  - Basic, but saves a lot of time

```
> bro -r trace-1.pcap tcp (output in conn.log)
```

<table>
<thead>
<tr>
<th>Time</th>
<th>Duration</th>
<th>Source</th>
<th>Destination</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>964953011</td>
<td>0.063756</td>
<td>10.20.12.187</td>
<td>207.126.127.69</td>
<td>http</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SrcPort</th>
<th>DstPort</th>
<th>Proto</th>
<th>SrcBytes</th>
<th>DstBytes</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>9002</td>
<td>80</td>
<td>tcp</td>
<td>0</td>
<td>?</td>
<td>RSTR</td>
</tr>
</tbody>
</table>

- Try for UDP and ICMP
Connection summaries (2)

- Connection states

<table>
<thead>
<tr>
<th>Connection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF</td>
<td>Normal establishment and termination</td>
</tr>
<tr>
<td>REJ</td>
<td>Connection attempt rejected</td>
</tr>
<tr>
<td>S0</td>
<td>Connection attempt seen, no reply</td>
</tr>
<tr>
<td>OTH</td>
<td>No SYN seen, partial connection</td>
</tr>
<tr>
<td>RSTO</td>
<td>Connection established, originator aborted</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Connection summaries (3)

- Fraction of connections with a given state?
  - TCP: SF, REJ, S0, OTH, RSTO
  - UDP: SF, REJ, S0, OTH, RSTO
  - ICMP: OTH
Weird activity

- Network traffic contains lots of weirdoes
  - Activity which does not conform to standard but is not an attack
  - Example: data being sent after RST

  > bro -r trace-1.pcap weird

- Scans

  > bro -r trace-1.pcap scan
Protocol analyzer

- Protocol-specific analysis
  - Log activity
  - Check for protocol-specific attacks
- Bro ships with analyzers for many protocols:
  - FTP, HTTP, POP3, IRC, SSL, DNS, NTP, ...
- Example: FTP analyzer

> bro -r trace-1.pcap ftp
> cat ftp.log
Packet filter

- Bro analyzes only the packets required by scripts’ analysis
  - Builds dynamically packet filter
- Seeing packet filter:
  > bro tcp ftp smtp print-filter
- Packet filter can be changed
- Bro skips whatever traffic does not match filters!
Dynamic protocol detection

- How does Bro know the analyzer for a connection?
- Default mechanism: examine the ports
- Problem: well-known ports are unreliable
- Bro can analyze protocols independent of ports
  - Dynamic protocol detection
  - Current support for HTTP, IRC, SMTP, SSH, FTP, POP3, BITTORRENT
  - Identifies potential protocol usage with signatures and then validates by parsing

```
bro -r trace-1.pcap -f "tcp" http-request http-reply dpd

bro -r trace-1.pcap -f "tcp" http-request http-reply bro-lite
```