Writing a TCP analysis expert system

...because it’s cool!

Jasper Bongertz
Airbus CyberSecurity
About me

- Working at Airbus CyberSecurity

- Network analysis & forensics since 2003
  - NetXRay, Sniffer Pro/Distributed, Clearsight
  - Ethereal since... uh... version 0.9something

- Creator of
  - TraceWrangler
  - blog.packet-foo.com
TCP Expert Systems
TCP Expert Systems

- Analyse all available TCP packets
- Diagnose various relevant symptoms
- Help the analyst find problems
Relevant? Uhm...
Okay, some „POLA“

- POLA – Principle Of Least Anstonishment:
  - it means that something behaves as expected

- What we want from an expert system:
  - useful symptoms
  - not getting swamped
  - rated by criticality
  - provide recommended fixes, if at all possible
Let's analyze some TCP

- **What's going on here?**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00000</td>
<td>192.168.122.69</td>
<td>192.168.101.111</td>
<td>TCP</td>
<td>62</td>
<td>2468 → 9887 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1</td>
</tr>
<tr>
<td>2</td>
<td>0.01152</td>
<td>192.168.101.111</td>
<td>192.168.122.69</td>
<td>TCP</td>
<td>62</td>
<td>9887 → 2468 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1349 SACK_PERM=1</td>
</tr>
<tr>
<td>3</td>
<td>0.01155</td>
<td>192.168.122.69</td>
<td>192.168.101.111</td>
<td>TCP</td>
<td>54</td>
<td>2468 → 9887 [RST] Seq=1 Win=0 Len=0</td>
</tr>
<tr>
<td>5</td>
<td>414.00758</td>
<td>192.168.101.111</td>
<td>192.168.122.69</td>
<td>TCP</td>
<td>62</td>
<td>[TCP Retransmission] 9887 → 2468 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK_PERM=1</td>
</tr>
</tbody>
</table>

- **And here?**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00000</td>
<td>172.18.0.122</td>
<td>172.18.50.1</td>
<td>TCP</td>
<td>74</td>
<td>51004 → 102 [SYN] Seq=0 Win=14600 [TCP CHECKSUM INCORRECT] Len=0 MSS=1460 SACK_PERM=1 Tsval=147472368 TSecr=0 WS=128</td>
</tr>
<tr>
<td>2</td>
<td>0.00275</td>
<td>172.18.50.1</td>
<td>172.18.0.122</td>
<td>TCP</td>
<td>70</td>
<td>102 + 51004 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 Tsval=2874599 TSecr=1072693248</td>
</tr>
<tr>
<td>3</td>
<td>0.00283</td>
<td>172.18.50.1</td>
<td>172.18.0.122</td>
<td>TCP</td>
<td>54</td>
<td>51004 → 102 [RST] Seq=1 Win=0 Len=0</td>
</tr>
<tr>
<td>4</td>
<td>1.40799</td>
<td>172.18.0.122</td>
<td>172.18.50.1</td>
<td>TCP</td>
<td>74</td>
<td>51010 → 102 [SYN] Seq=0 Win=14600 [TCP CHECKSUM INCORRECT] Len=0 MSS=1460 SACK_PERM=1 Tsval=147472720 TSecr=0 WS=128</td>
</tr>
<tr>
<td>5</td>
<td>1.41086</td>
<td>172.18.50.1</td>
<td>172.18.0.122</td>
<td>TCP</td>
<td>70</td>
<td>102 + 51010 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 Tsval=2874740 TSecr=1095761920</td>
</tr>
<tr>
<td>6</td>
<td>1.41093</td>
<td>172.18.0.122</td>
<td>172.18.50.1</td>
<td>TCP</td>
<td>51</td>
<td>51010 + 102 [RST] Seq=1 Win=0 Len=0</td>
</tr>
</tbody>
</table>
Investigating file "TCPSample01.pcapng"

| Address A;Port A;Address B;PortB;Handshake State;Teardown Client;Teardown Server;Reset State;Symptoms;State Location;Client IP;First IP seen;Client determined;Common MSS;GapRatio;Total Symptom Score;Initial RTT (ns);SYN Packets Seen;SYN Reset count;Client FIN count;Server FIN count;Client RST count;Server RST count;Client packet count;Server packet count;Client discarded packets;Server discarded packets;Client CRC errors;Server CRC errors;Client TTL;Server TTL;Client MSS;Server MSS;Display Filter;Capture Filter |
| 192.168.0.102;49230;23.235.37.194;80;Complete;None;None;Server;tsRoutingDuplicate|tsSYNACKRetransmit|tsPacketsAfterReset|tsMultiServerTTL;ClientLocal;192.168.0.102;192.168.0.102;true;1460;63;46;11529000;1;0;0;0;2;3;6;2;0;3;0;128;60;1460;1460;ip.addr==192.168.0.102 and ip.addr==23.235.37.194 and tcp.port==49230 and tcp.port==80;host 192.168.0.102 and 23.235.37.194 and tcp port 49230 and 80 |
Challenges

• **TCP analysis is easy to learn, hard to master:**
  • it requires experience, experience, experience
  • Example: „Hey, a RESET Packet! Is that bad?“

• **Challenge: Programmers writing analysis software may not be TCP experts:**
  • „I guess **ACK too long** is a useful symptom, right?“
  • BTW: nope. About the same as **Window Frozen** 😁
Challenges

- There are three truths in TCP:
  - What the TCP client knows
  - What the TCP server knows
  - What the trace contains (notice: no "knows" here)

- Key factor: the capture location and setup:
  - timings, packet out-of-order situations
  - packet loss, duplicate frames
Attention – 4 Animations ahead!
Three Truths Example #1

Packet Sent

Packet Seen

Packet Received
Three Truths Example #2

Packet Sent

Capture Laptop

Packet Seen

Nothing arrived
Three Truths Example #3

Packet Sent

Capture Laptop

Nothing arrived

Server

Nothing arrived
Three Truths Example #4

Packet Sent

Capture Laptop

Packet dropped

Packet Received
The 6-Tuple
Huh?! 6-Tuple???

- TCP analysis requires looking at a single conversation at a time
- Usually the 5-Tuple is enough:
  - Protocol (UDP/TCP)
  - Source IP, DestinationIP
  - Source Port, DestinationPort
- New: ISN (Initial Sequence Number)
TCP Handshake
TCP handshake

- Very important: the TCP handshake packets
  - TCP options
  - Who’s who
  - Initial Rout Trip Time
- Challenge: determining handshake packets
- What if we don’t have any? Or just some? Or they arrive out of order?
TCPInvestigator is a prototype

- Mostly focused on the handshake/teardown as of now
  - Investigating the full conversation flow still needs to be added
  - Has 102 different data points already
- A full rewrite may be necessary at a later stage
  - Nested case statements are not very elegant
  - Maybe going for a state machine instead
// Check for sequence number wraps about to happen

if Parser.NextExpectedSequenceNumber.WrapCount = (Parser.SequenceNumber.WrapCount + 1) then
    State.Symptoms := State.Symptoms + [tsttcpSequenceWrapped];

if (tcpSYN in Parser.Flags) then
begin
    // Got a SYN or SYN/ACK in the new packet
    // TODO: may need to update TCPTuple ISN via UpdateISN

    case State.Handshake of
        hsNone

        begin
            // we had packets, but neither SYN nor SYN/ACK yet. This is very uncommon and
            State.Symptoms := State.Symptoms + [tsHandshakeOutOfOrder];

            if SourceIsClient then
                begin
                    if Timestamp < State.LastClientTimestamp then
                        State.Symptoms := State.Symptoms + [tsFrameOutOfOrder];
                end
            else
                begin
                    if Timestamp < State.LastServerTimestamp then
                        State.Symptoms := State.Symptoms + [tsFrameOutOfOrder];
                end;
        end;
    end;
1 Pass? 2 Pass? 3 Pass?

- Examining TCP packets in a single pass may not be enough
  - depends on capture file size and resources
  - E.g. “how long can you wait for a possible out-of-order/retransmission segment“?
- I expect at least 2 passes are required
  - e.g. also to deal with SACK edge blocks
Symptom detection

- **Single packet**
  - e.g. missing TCP options like MSS in the handshake

- **Conversation packets**
  - this is the majority, e.g. packet loss

- **Global**
  - e.g. pulling info from other conversations as baseline, like iRTT
Demos!!!11
Q&A

Mail: jasper@packet-foo.com
Web: blog.packet-foo.com
Twitter: @packetjay