TCPtransform
(Offline version of TCPopera)

Seung-Sun (Gary) Hong, S. Felix Wu

Security Laboratory
Computer Science Department
University of California - Davis
Outlines

- Motivations
- Related work
- TCPtransform/TCPopera development
- Design & Implementation
- Validation tests
  - TCPtransform
    - FTP traffic reproduction
  - TCPopera
    - Interactive traffic reproduction, IPS testing
- Conclusions & Future directions
Motivations

- Industrial request
  - Having test traffic for security products
  - In-line device testing, e.g. IPS, firewall, router

- Internal request
  - Replaying traffic captured from MINOS honeypot on DETER
  - UC Davis is one of major participants in DETER project which is a large-scale network emulation environment for security protocol/product testing.
Motivations

- Limitation of conventional trace replay tools
  - Not capable of stateful emulation of TCP connections
  - Inconsistent data/control packets generation
    - E.g. generation of ghost packets
  - No good for in-line device testing such as IPS testing
- Live security test environments require
  - Realistic test traffic and packet contents
  - more interactive traffic replay approach
Related work

- **Trace-based traffic replaying**
  - Easy to implement and mimic system behaviors
  - Real traffic, sufficient diversities
  - Hard to adjust trace for various test conditions
    - Assuming the test condition is the same as the time at the trace was recorded

- **Analytic-model based traffic generation**
  - Easy to control/adjust traffic generation models
  - Statistically identical to traffic models.
  - Hard to support trace contents for security test environments
Trace-based traffic replaying

- **TCPReplay/Flowreplay**
  - Static trace replaying mainly for NIDS testing
  - Flowreplay is a TCP client emulator

- **TCPivo**
  - High-performance trace replay tool
  - I/O management, Timer accuracy, Null-padding payload

- **Monkey**
  - HTTP emulator for a Google server
  - Monkey See for TCP tracing, Monkey do for TCP replaying

- **Tomahawk**
  - A tool for testing in-line blocking capabilities (IPS)
  - Operable across layer-2 connection.
TCPtransform/TCPopera development

Milestone of TCPopera development

Details
- Requirements, Architecture
- Implementation of Major Components
- Reproductivity Snort testing
- Inter-connection dependency models, DETER deployment

Time
- Design: 03/04
- 1st development phase: 06/04
- Evaluation: 09/04
- 2nd development phase: 01/05
- 04/05
- 09/05
- 12/05

Output
- TCPtransform
- TCPopera
- DIMVA’05
- RAID’05
Property-oriented trace replaying

- Extract traffic parameters from Input trace records through the reverse-engineering
- Adjust traffic parameters according to test conditions
- Feed new traffic parameters to input packet sequence

Traffic Property Engine

Input TCPdump file → Traffic Parameters → Traffic Configuration

Packet Sequence → Event → Packet Processing → Output TCPdump file

Feedback
TCPtransform Components

- TCPopera/TCPtransform Major Components

TCPtransform components

- Configuration Files
  - Traffic Models
  - Flow Preprocess
  - Packet Injection
  - TCP Functions
  - IP Flow Process
  - Packet Capturing

TCPopera Components

- TCPopera Timer
- TCPopera Control

Physical link
TCPtransform Components

- **Flow Preprocess**
  - Preparing IP flows
  - Extraction of TCP connection and IP flow parameters
    - RTT, transmission rate, packet loss rate, path MTU
  - Address remapping, ARP emulation

- **IP Flow process**
  - Creating a POSIX thread for each IP flow
  - TCP control block emulation

- **Traffic Models**
  - TCP parameters for the initiation of TCP control blocks
  - Gap-based packet loss model
TCPtransform Components (Cont’d)

- **TCP Functions**
  - Based on BSD4.4-Lite release (1994) - TCP Reno
  - 8 TCP timers
    - Slow timer (500ms), Fast timer (200ms)
  - Timeout & Retransmission
    - RTT measurement
  - Fast Retransmit & Fast Recovery
  - Flow & Congestion Control

- **Packet Injection/Packet Capturing**
  - Libnet and Pcap
  - IP/TCP checksum recalculation if a packet is modified
TCPtransform Validation

- FTP traffic reproduction
  - Imitating a FTP connection to download a 8M file from 3 different public GNU servers
  - Sampled over 10,000 FTP connections from each server

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Host name (IP address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berlin</td>
<td>German</td>
<td>ftp.cs.tu.berling.de (130.149.17.12)</td>
</tr>
<tr>
<td>NCTU</td>
<td>Taiwan</td>
<td>ftp.nctu.deu.tw (140.113.27.181)</td>
</tr>
<tr>
<td>Charlie</td>
<td>Sweden</td>
<td>ftp.chl.charlmers.se (129.16.214.70)</td>
</tr>
</tbody>
</table>

- Test setup
  - Collect an input tcpdump file from a local FTP server.
    - To remove any noise, we directly connect the client machine to the FTP server.
  - TCPtransform reproduced FTP connections for each server
TCPtransform Validation

- Gab-based Packet Loss Model

**State 1**
Packet received successfully

**State 2**
Packet lost within a burst

**State 3**
Packet received within a burst

$p_{11}$

$p_{12}$

$p_{22}$

$p_{23}$

$p_{33}$

$p_{32}$

If $n > N$

N: Maximum number of packets in Packet loss period,
n: number of packets in Packet loss period
**Q distribution**

- $\chi^2$-like test to compare the similarity between long-term and short-term profile.
- Partition sample space $S$ into $bin_i$.
- $N$: Total number of events
- $Y_i$: Number of event occurrences for $bin_i$.
- $P_i$: Probability of event occurrences for $bin_i$ ($Y_i/N$)
- $Y'_i$ and $N'$ for short-term profile.

$$Q = \sum_{i=1}^{k} \frac{(Y'_i - N' \times P_i)^2}{N \times P_i}$$
# Test conditions

- **4 traffic variables**

<table>
<thead>
<tr>
<th>Server</th>
<th>Berlin</th>
<th>NCTU</th>
<th>Charlmers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet loss rate</td>
<td>0.0003</td>
<td>0.0002</td>
<td>0.0001</td>
</tr>
<tr>
<td>Loss burst size (Pareto)</td>
<td>Shape</td>
<td>1.1</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Packet burst size (Pareto)</td>
<td>Shape</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>RTT (msec)</td>
<td>Avg.</td>
<td>152</td>
<td>260</td>
</tr>
<tr>
<td></td>
<td>Stdev</td>
<td>9.161</td>
<td>14.881</td>
</tr>
</tbody>
</table>
Test results

- NPR (Number of Packet Reordering)

Berlin, German

NCTU, Taiwan

Charlmers, Sweden

Distribution of NPR (Berlin)

Distribution of NPR (NCTU)

Distribution of NPR (Charlmers)

Q Distribution of NPR (Berlin)

Q Distribution of NPR (NCTU)

Q Distribution of NPR (Charlmers)
Test results (cont’d)

Session Duration

Berlin, German

NCTU, Taiwan

Charlmer, Sweden

Distribution of session duration (Berlin)

Distribution of session duration (NCTU)

Distribution of session duration (Charlmer)

Q Distribution of session duration (Berlin)

Q Distribution of session duration (NCTU)

Q Distribution of session duration (Charlmer)
TCPopera Validation

Test setup

TCPopera nodes
- 2 GHz Intel Pentium 4, 768MB RAM
- Internal: Redhat 8 (2.4.18), External: Redhat 9 (2.4.20)

Network Emulator
- 455MHz Pentium II Celeron, 256MB RAM
- FreeBSD5.0, IPFW (with Dummynet)

Snort 2.3
- 3.2 GHz Intel Pentium 4 Processor, 512MB
- Slackware 10.0 (2.4.26)
- All Snort rules are enabled including the Stream4 analysis
TCPopera traffic reproduction

- DARPA IDEVAL99 (first 12 hours of 03/29/99)

<table>
<thead>
<tr>
<th>Category</th>
<th>Input trace</th>
<th>TCPopera</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No loss</td>
</tr>
<tr>
<td>IP</td>
<td>Packets</td>
<td>1,502,584</td>
</tr>
<tr>
<td></td>
<td>Bytes</td>
<td>234,434,486</td>
</tr>
<tr>
<td>TCP</td>
<td>Packets</td>
<td>1,225,905</td>
</tr>
<tr>
<td></td>
<td>Bytes</td>
<td>194,927,209</td>
</tr>
<tr>
<td>UDP</td>
<td>Packets</td>
<td>276,286</td>
</tr>
<tr>
<td></td>
<td>Bytes</td>
<td>39,474,602</td>
</tr>
<tr>
<td>ICMP</td>
<td>Packets</td>
<td>393</td>
</tr>
<tr>
<td></td>
<td>Bytes</td>
<td>32,675</td>
</tr>
<tr>
<td>TCP connections replayed</td>
<td>18,138</td>
<td>18,138</td>
</tr>
<tr>
<td>TCP connections completed</td>
<td>14,974</td>
<td>14,971</td>
</tr>
</tbody>
</table>
TCPopera traffic reproduction

- Traffic volume comparison (every minute)

![Graph of IP Bytes and TCP Bytes over time](image-url)
TCPopera validation (Snort Evaluation)

- **ITRI Dataset**
  - Collected for 30 minutes from a host within 140.96.114.0/24 segment in Taiwan
  - Major applications: HTTP, P2P (eDonkey), FTP

- **Evaluation results**

<table>
<thead>
<tr>
<th>Signature</th>
<th>No. of alerts</th>
<th>Input trace</th>
<th>TCPopera</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>No-loss</td>
</tr>
<tr>
<td>ICMP Destination/Port Unreachable</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>ICMP Destination/Host Unreachable</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ICMP Destination Unreachable Fragmentation needed but DF bit is set</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P2P eDonkey Transfer</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>(stream4) Possible retransmission detection</td>
<td>38</td>
<td>212</td>
<td>200</td>
</tr>
<tr>
<td>(stream4) WINDOW violation detection</td>
<td>488</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>537</td>
<td>226</td>
<td>212</td>
</tr>
</tbody>
</table>
Snort Evaluation – stream4 analysis

- Possible retransmission detection
  - Detecting an attempt to packet replaying attack
  - TCPopera’s delayed ACKs confused the stream4 re-assembler.

- WINDOW violation detection
  - Detecting an attempt to write the outside of the receiver’s window.
  - Mishandling of incomplete TCP connections.
    - Mistakenly assume the connection is established.
  - Strict rules on handling RST segments.
    - No resetting TCP connection, instead update the window size an RST segment is carrying.
Conclusions

- TCPopera does Interactive trace replaying with a stateful emulation of TCP connections.
- Initial evaluation showed a positive sign in the usefulness of TCPopera.
- Providing more methodologies for the security product evaluation.
- Deployable in a large-scale emulation environment like DETER.
- TCPopera is an on-going project.
Future directions

- Next TCPopera development phase
  - Porting TCPopera into DETER environment.
  - More in-line devices evaluation such as IPS.
    - Adding more evasive techniques for IPS testing
  - Supporting more application-specific inter-connection dependency models
  - Adding more TCP/UDP traffic models
  - Adding a centralized TCPopera GUI to control multiple TCPopera nodes.
Thanks & Questions

- Many thanks for paying attention to the talk.
- Any question