DeltaShaper
Enabling Unobservable Censorship-resistant TCP Tunneling over Videoconferencing Streams

Diogo Barradas          Nuno Santos          Luís Rodrigues

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
Censors monitor / control Internet access
Censors monitor / control Internet access
Censors attempt to block covert channels
DeltaShaper

• **Goals**
  - Establish a covert TCP/IP channel
  - Maintain unobservability
  - Resist against network perturbations
## Multimedia protocol tunneling

### Security

<table>
<thead>
<tr>
<th>System / Properties</th>
<th>Active/Passive Attack Resistance</th>
<th>Arbitrary Data Transmission</th>
<th>Interactive Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeWave</td>
<td>-</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><em>Houmansadr et al.</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio Modulation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Facet</td>
<td>✔</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><em>Li et al.</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video Embedding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CovertCast</td>
<td>✔</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td><em>McPherson et al.</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video Modulation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeltaShaper</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><em>Video Modulation</em></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Threat model

• **Assumptions:**
  - Packets carrying multimedia data are encrypted

• **Censor’s Capabilities:**
  - *Deep Packet Inspection*
  - Observe, store and analyze traffic flows
  - Apply artificial constraints on the network

• **Censor’s Limitations:**
  - Unable to decipher the content of Skype packets
  - Not in collusion with the video-conferencing provider
  - Attempts to minimize collateral damage
A naïve approach at data modulation

- Replace chat video frames
- Encode data in all available pixels

$1\text{px} = 24\text{b}$
$R = 8\text{b}$
$G = 8\text{b}$
$B = 8\text{b}$

$\sim 922\text{ kB / frame}$
Drawbacks of naïve data modulation

• **Data loss**
  • Lossy compression (downsampling + quantization)

• **Abnormal traffic patterns**
  • Poor compression (spatial & inter-frame redundancy)

1px = 24b
R = 8b
G = 8b
B = 8b

480 px

640 px

~922 kB / frame
C1: Can we distinguish regular from irregular Skype streams?

- Traffic signatures appear to be different
  - Packet lengths frequency distribution

Frames change extensively

Frames do not change
C2: How much throughput can we achieve while preserving unobservability?
C3: How to maintain unobservability in adverse network conditions?
Contributions

• **DeltaShaper**: A censorship-resistant system
  • Tunnel TCP/IP data over Skype videocalls

• **Distinguish regular / irregular Skype call streams**
  • Packet frequency distribution / EMD

• **Maximize throughput and maintain unobservability**
  • Explore the space encoding parameters

• **Adaptation to network conditions**
  • Dynamic calibration of encoding parameters
How to characterize Skype streams?

• **Characteristic Function - Create a stream signature**
  • Frequency distribution of packet lengths

• **Similarity Function - Quantify streams’ differences**
  • *Earth Mover’s Distance* (EMD)
Different videos generate distinct traffic

- Differences between signatures can be quantified
  - *Earth Movers’ Distance*
Different videos generate distinct traffic

- Censors can identify streams with unusual traffic
Can we encode data and maintain unobservability?

- **Strawman**: Embed a small payload in each frame
- **Generated traffic does not reflect this embedding**
A better approach for data modulation

- **Strive for unobservability**
- **Accommodate for lossy compression**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_p)</td>
<td>payload frame area (pixel\times\text{pixel})</td>
</tr>
<tr>
<td>(a_c)</td>
<td>cell size (pixel\times\text{pixel})</td>
</tr>
<tr>
<td>(b_c)</td>
<td>color encoding (bits)</td>
</tr>
<tr>
<td>(r_p)</td>
<td>payload frame rate (frames/s)</td>
</tr>
</tbody>
</table>
Adapt to network conditions

- **Calibrate encoding parameters**
  - Maintain unobservability
  - Modulate max. amount of data
DeltaShaper adaptation mechanism

- **Periodically:**
  - Estimate network conditions from recorded baselines
  - Select adequate parameters from pre-computed table

Which set is closest?

Carrier signature

Encoding parameters

<table>
<thead>
<tr>
<th>Cond. 1</th>
<th>Cond. 2</th>
<th>Cond. n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap 1</td>
<td>ap 2</td>
<td>ap n</td>
</tr>
<tr>
<td>ac 1</td>
<td>ac 2</td>
<td>ac n</td>
</tr>
<tr>
<td>bc 1</td>
<td>bc 2</td>
<td>bc n</td>
</tr>
<tr>
<td>rp 1</td>
<td>rp 2</td>
<td>rp n</td>
</tr>
</tbody>
</table>
Implementation challenges

• **Network interaction**
  • Allow transparent TCP/IP communication

• **Video processing**
  • Combine carrier / payload frames

• **Video-conferencing software as a *black-box***
  • Send covert frames without modifying Skype
DeltaShaper client module

Client Endpoint

- Network Namespace
  - Client Application
    - VETH0
      - 10.10.10.11
    - VETH1
      - 10.10.10.10

- Encoder Adapter
  - Payload Encoder
    - Payload Queue
      - Frame Queue
      - Payload Streamer
        - Stream Blender (Snowmix)
  - Carrier Frame

- Virtual Camera
  - /dev/video0

- FFMPEG

Covert Stream
DeltaShaper server module
Evaluation Steps

1. Can we distinguish Skype streams?

2. Can we balance throughput and unobservability?

3. How well does DeltaShaper perform?
Can we distinguish Skype streams?

- 83% accuracy in distinguishing Skype streams
- DeltaShaper streams must remain under $\Delta I$
Can we balance throughput and unobservability?

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_p)</td>
<td>payload frame area (pixel(\times)pixel)</td>
<td>320 x 240</td>
</tr>
<tr>
<td>(a_c)</td>
<td>cell size (pixel(\times)pixel)</td>
<td>8 x 8</td>
</tr>
<tr>
<td>(b_c)</td>
<td>color encoding (bits)</td>
<td>6</td>
</tr>
<tr>
<td>(r_p)</td>
<td>payload frame rate (frames/s)</td>
<td>1</td>
</tr>
</tbody>
</table>
How well does DeltaShaper perform?

• **Achieved configuration:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap</td>
<td>payload frame area (pixel×pixel)</td>
<td>320 x 240</td>
</tr>
<tr>
<td>ac</td>
<td>cell size (pixel×pixel)</td>
<td>8 x 8</td>
</tr>
<tr>
<td>bc</td>
<td>color encoding (bits)</td>
<td>6</td>
</tr>
<tr>
<td>rp</td>
<td>payload frame rate (frames/s)</td>
<td>1</td>
</tr>
</tbody>
</table>

• **Performance**
  • Raw throughput: **7.2 Kbps**
  • Round-Trip-Time: **2s 973ms**
How well does DeltaShaper perform?

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Protocol Session W/ DS (mm:ss)</th>
<th>Protocol Session W/o DS (mm:ss)</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wget (4kB file)</td>
<td>0:22</td>
<td>&lt; 0:01</td>
<td>3,142.9 x</td>
</tr>
<tr>
<td>FTP (4kB file)</td>
<td>1:43</td>
<td>0:09</td>
<td>11.4 x</td>
</tr>
<tr>
<td>SSH + SMTP</td>
<td>2:41</td>
<td>0:38</td>
<td>4.2 x</td>
</tr>
<tr>
<td>SSH</td>
<td>1:29</td>
<td>0:06</td>
<td>14.8 x</td>
</tr>
<tr>
<td>Telnet</td>
<td>1:13</td>
<td>0:06</td>
<td>12.2 x</td>
</tr>
<tr>
<td>Netcat chat</td>
<td>0:01</td>
<td>&lt; 0:01</td>
<td>166.7 x</td>
</tr>
<tr>
<td>SSH Tunnel</td>
<td>2:19</td>
<td>0:22</td>
<td>6.3 x</td>
</tr>
</tbody>
</table>

- DeltaShaper allows for the execution of traditional TCP/IP applications which cover different users’ needs.
Conclusions

- **DeltaShaper: A censorship-resistant system**
  - Supports high-latency / low-throughput TCP applications

- **Maximize throughput and preserve unobservability**
  - Greedy exploration of encoding configurations

- **Adaptation in multimedia protocol tunneling**
  - Provides improved unobservability
  - Could also enhance similar systems

http://web.ist.utl.pt/diogo.barradas