Background

- Detection Problem
  - Overlay Network
  - Public-Key Encryption & RC-4
  - Control, Bursty, and Real-time Media combined
  - Recent Advances

- Inhibition Problem
  - NAT-Piercing
  - Hiding on ports 80, 443
  - Intelligent & Reactive
Questions

- Given some limited insight into Skype traffic, how can we inhibit it?
- How much can we play with traffic before it resorts to “Skype fail-over mode”? 
Test Environment

- My Skype client
  - PC behind Linux-based NAT
  - Symantec client-side firewall provides logs of all Skype network activity
  - IPTABLES-based NAT provides loss, latency, reordering simulations

- Remote Hosts
  - Skype Call Test Service
  - Equivalent host behind equivalent NAT
Qualitative Usability Index

- 10 Ideal.
- 9 Excellent. (Normal)
- 8
- 7 Fair.
- 6
- 5 Understandable, but undesirable.
- 4
- 3 Cannot get through a single sentence.
- 2
- 1 No communication.
Test 1: Drop Packets

- Longer time to log in
- Moved to TCP
- Kept trying UDP
- Able to connect to Skype
- Able to make calls
- Quality 9.
Test 2: Drop Some Packets

- Drop one in 10 packets
- Switched to random with \( p=0.1, 0.2, \ldots 0.5 \)
- Still able to sign on (used TCP flows)
- Able to make calls
- Quality of calls degraded
- No fail-over to TCP
# Test 2 Results

<table>
<thead>
<tr>
<th>Condition</th>
<th>Usability Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:10, Skype Test Call</td>
<td>8.5A</td>
</tr>
<tr>
<td>1:10, Video conversation</td>
<td>8.5A, 9V</td>
</tr>
<tr>
<td>p = 0.1</td>
<td>8.5A, 9V</td>
</tr>
<tr>
<td>p = 0.2</td>
<td>7A, 8V</td>
</tr>
<tr>
<td>p = 0.3</td>
<td>5A, 7V</td>
</tr>
<tr>
<td>p = 0.4</td>
<td>3A, 6V</td>
</tr>
<tr>
<td>p = 0.5</td>
<td>Failed over to TCP</td>
</tr>
</tbody>
</table>
Test 3: Drop Some Packets

- Random with $p=0.1, 0.2, \ldots$
- Applied to entire network connection
- Able to sign on
- Able to make calls… sometimes
- Quality of calls degraded more quickly
# Test 3 Results

<table>
<thead>
<tr>
<th>Condition</th>
<th>Usability Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>p = 0.1</td>
<td>8.5</td>
</tr>
<tr>
<td>p = 0.2</td>
<td>4</td>
</tr>
<tr>
<td>p = 0.3</td>
<td>1</td>
</tr>
</tbody>
</table>
Test 4: Introduce Latency

- Able to sign in
- Able to make calls
- Noticeable latency (obviously)
- 100-1000 ms: Did not affect call quality
- 3000ms +: Problems started
  - Sometimes connected w/o audio, video
  - Calls dropped
# Test 4 Results

<table>
<thead>
<tr>
<th>Condition</th>
<th>Usability Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>100ms</td>
<td>9</td>
</tr>
<tr>
<td>300ms</td>
<td>9</td>
</tr>
<tr>
<td>1000ms</td>
<td>9</td>
</tr>
<tr>
<td>3000ms</td>
<td>1…9</td>
</tr>
<tr>
<td>5000ms</td>
<td>1</td>
</tr>
</tbody>
</table>
Test 5: Duplication

- No effect
  - Tried 50% and 100%
- Able to log in
- Able to make calls
- Quality 9
Test 6: Corruption

- Similar to network-level drop results
- Even 1% noticeable impact...
- Able to sign in
- Able to make calls... sometimes
- Quality degraded quickly
## Test 6 Results

<table>
<thead>
<tr>
<th>Condition</th>
<th>Usability Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>8</td>
</tr>
<tr>
<td>10%</td>
<td>6.5</td>
</tr>
<tr>
<td>20%</td>
<td>5</td>
</tr>
<tr>
<td>30%</td>
<td>Could not connect</td>
</tr>
</tbody>
</table>
Test 7: Re-ordering

- Delay $p$ of the packets by $t$ ms
- $p \in \{20\%, 40\%, 50\%\}$
- $t \in \{10, 30, 100\}$
- Only high delay had impact
- Able to connect
- Able to make calls
- Only affected call quality
## Test 7 Results

<table>
<thead>
<tr>
<th></th>
<th>10ms</th>
<th>30ms</th>
<th>100ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>20%</td>
<td>8.5</td>
<td>8.5</td>
<td>8</td>
</tr>
<tr>
<td>40%</td>
<td>8.5</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>50%</td>
<td>9</td>
<td>9</td>
<td>6</td>
</tr>
</tbody>
</table>
Conclusions

- Some things entirely useless
- Fighting randomness with randomness is best approach
  - Skype acts predictably when it feels safe
  - Avoiding blocking consistent false positives from imperfect heuristics
  - Consistent adverse conditions can be detected and worked around
Injecting Traffic would not work

Network-level tests: Might be useful to change traffic characteristics during conversation
Resources

- **Background**
  - Silver Needle in the Skype (pentesting)
    - Biondi & Desclaux, BlackHat Europe ’06
  - Revealing Skype traffic: When randomness plays with you
    - Bonfiglio et al, SIGCOMM ’07

- **Tools**
  - Netem (linux-foundation.org)
  - Iptables (netfilter.org)