Slitheen++:
Stealth TLS-based Decoy Routing

Benedikt Birtel, Christian Rossow
Introduction Decoy Routing

Client

Browser

Censor Router

censored.com

uncensored.net

Censor Domain
Introduction Decoy Routing

1. Establish connection to censored.com
Introduction Decoy Routing

1. Establish connection to censored.com
2. Censor denies access

Browser

Client

Censor Router

censored.com

uncensored.net
Introduction Decoy Routing

Client

Browser

Censor Domain

censored.com

uncensored.net

Censor Router
Introduction Decoy Routing
Introduction Decoy Routing

Censor Domain

Browser

Decoy Routing
Client

Censor Router

Client

censored.com

Decoy Router

uncensored.net
Introduction Decoy Routing

1. Establish a connection to censored.com via decoy software

Censor Domain

Browser

Decoy Routing Client

Client

Censor Router

Decoy Router

censored.com

uncensored.net
Introduction Decoy Routing

2. Decoy client establishes a tagged connection to uncensored.net, including the real target of the connection.
Introduction Decoy Routing

3. The decoy router detects the tag and reroutes the data to censored.com

4. Data received from the real target will be redirected back to client, using the IP information of uncensored.net
Introduction Decoy Routing

Browser

Decoy Routing Client

Advanced Censor with Traffic Analysis

Decoy Router

censored.com

uncensored.net

Censor Domain

Slitheen++ - benedikt.birtel@cispa.saarland
Introduction Decoy Routing

Browser

Decoy Routing Client

The traffic analysis reveals the decoy routing

Fingerprints do not match

Advanced Censor with Traffic Analysis

Decoy Router

censored.com

Client

uncensored.net

Censor Domain
Slitheen

Browser

Advanced Censor with Traffic Analysis

Client

censored.com

uncensored.net

Censor Domain
Slitheen++ - benedikt.birtel@cispa.saarland

Censor Domain

Browser

Overt User Simulator

Client

Advanced Censor with Traffic Analysis

censored.com

uncensored.net
Slitheen++ - benedikt.birtel@cispa.saarland

Censor Domain

Browser

Overt User Simulator

Client

Advanced Censor with Traffic Analysis

censored.com

Overt Target

uncensored.net

Legends

HTTPS Overt Connection
Covert browser establishes a connection to the OUS using SOCKS5.
Covert Browser

(De-) Multiplexer

Overt User Simulator

Client

Advanced Censor with Traffic Analysis

Transmissions between OUS and relay are now carrying multiplexed traffic.

Covert upload has been appended to overt requests.

Legends

HTTPS Overt Connection
Covert Connection
Multiplexed Connection

Overt Target
uncensored.net
censored.com
Slitheen

Censor Domain

- Covert Browser
  - (De-) Multiplexer
    - Overt User Simulator
      - Client

- Advanced Censor with Traffic Analysis

- Relay Station
  - (De-) Multiplexer
    - Covert Target
censored.com

- Overt Target
uncensored.net

Legend:
- HTTPS Overt Connection
- Covert Connection
- Multiplexed Connection

Relay extracts covert upload, sends it to covert target.
If covert downstream data is available, it replaces “leaf content” (e.g. pictures) of the overt connection.
Slitheen++ - Improvements: Stealth Upload

- Slitheen:
  - Appends covert upload data to overt requests completely
  - This changes the overt upload behavior such that a censor could detect it

- Slitheen++:
  - (De-) Compresses HTTP headers on OUS and relay station like HTTP/2
  - Upload data is inserted into the space gains
  - Overt upload packet sizes are not changed by multiplexing covert data into it
Slitheen++ - Improvements: Scheduling

- **Slitheen:**
  - Used a single queue to store application data from all covert connections and send them in the order they have been received
  - This lead to problems:
    - Starvation of connections
    - Lower performance due to aggressive DDoS protection mechanisms by covert targets

- **Slitheen++:**
  - Implements a queue per covert connection and store its data in it
  - The connection are scheduled based on:
    - First-In-First-Out (FIFO) strategy
    - Sliced Round-Robin (SRR) strategy
Slitheen++ - Improvements: Thinking Time

- Slitheen:
  - OUS loads a web page without any break between two loads
  - This creates a “constant” covert goodput capability
  - Unusual browsing behavior that may will be detected by a censor

- Slitheen++:
  - Added a Thinking Time (TT) implementation between page loads
  - TT creates a chainsaw shaped covert goodput pattern
  - Less aggressive browsing pattern
Slitheen++ - Improvements: Crawling

- **Slitheen:**
  - The OUS did only load a single website in a loop
  - Is likely to be detected by a censor

- **Slitheen++:**
  - Implements crawling by extracting links to other overt pages from the currently loaded one
  - Is less likely to be detected by a censor since the user is not always visiting the same page over and over again
10 domains, 3 web sites from each domain (scenario)

We used different setups with various parameters such as TT and Crawling en-/disabled

Repeated the measurement 10 times for the final results

We measured:
- User experience (web site loading time)
- Covert Goodput
- Overt forwarding cost
- Slitheen++ decreases user experience compared to the baseline
- Crawling can have a positive effect
- TT is performing worse than crawling
- Crawling with TT combines the advantages and disadvantages of both
Slitheen++ - Evaluation: Covert goodput
Slitheen++ - Evaluation: Overt forwarding cost

<table>
<thead>
<tr>
<th>Packet Delay D</th>
<th>Down Avg (D) %</th>
<th>Down Max (D)</th>
<th>Setups involved %</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 1 ms</td>
<td>0.0029</td>
<td>0.0060</td>
<td>95</td>
</tr>
<tr>
<td>&gt; 5 ms</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0</td>
</tr>
<tr>
<td>&gt; 10 ms</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0</td>
</tr>
<tr>
<td>&gt; 30 ms</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0</td>
</tr>
<tr>
<td>&gt; 50 ms</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Packet Delay D</th>
<th>Up Avg (D) %</th>
<th>Up Max (D) %</th>
<th>Setup involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 1 ms</td>
<td>0.0149</td>
<td>0.0248</td>
<td>100</td>
</tr>
<tr>
<td>&gt; 5 ms</td>
<td>0.0035</td>
<td>0.0090</td>
<td>90</td>
</tr>
<tr>
<td>&gt; 10 ms</td>
<td>0.0033</td>
<td>0.0090</td>
<td>90</td>
</tr>
<tr>
<td>&gt; 30 ms</td>
<td>0.0018</td>
<td>0.0036</td>
<td>80</td>
</tr>
<tr>
<td>&gt; 50 ms</td>
<td>0.0007</td>
<td>0.0007</td>
<td>15</td>
</tr>
</tbody>
</table>

Threads competed for resources, causing delays, especially for the upstream direction
Limitations/Future Work

- Only CPU-based implementation
  - PISA implementation to scale up and increase the bandwidth
- Thinking time and crawling decrease the user experience
  - Better, covert load dependent crawling as a fix
- Usage of video streaming or gaming as overt application
Thank you for your attention

Benedikt Birtel
E-Mail: benedikt.birtel@cispa.saarland

Christian Rossow
E-Mail: rossow@cispa.saarland