ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship

Philipp Winter\textsuperscript{1}, Tobias Pulls\textsuperscript{1}, and Jürgen Fuß\textsuperscript{2}

\textsuperscript{1}Karlstad University \textsuperscript{2}FH Hagenberg

November 4, 2013
Using Tor in China

Tor bridge

DPI box

Tor user
GFW actively probes bridges!
...and blocks their IP:port tuple
Let’s make active probing useless!
ScrambleSuit in a nutshell

- Censorship-resistant **polymorphic** transport protocol.
- Relies on secret which is shared **out-of-band**.
- Disguises Tor’s flow properties.
- **Maximise throughput** while aim for acceptable level of obfuscation!
The Big Picture

Other obfsproxy modules: **obfs2** and **obfs3**.
Thwarting active probing

- Client must prove knowledge of shared secret in **first** message.

- ...otherwise, the server remains silent.

- Two mechanisms: **Uniform Diffie-Hellman** and **session tickets**.

- Session ticket is always issued after successful authentication.

- Bridge does not disguise aliveness!
Authenticated uniform Diffie-Hellman

Client

\[ X \| P_C \| M_C \| \text{MAC}_{k_B}(X\|P_C\|E) \]

\[ Y \| P_S \| M_S \| \text{MAC}_{k_B}(Y\|P_S\|E) \]

\[ \text{Enc}_{k_t}(k_{t+1} \| T_{t+1}) \]

\[ \text{handshake complete} \]

\[ \text{Enc}_{k_t}(\text{Tor traffic}) \]

Server

Legend:
- \( X \): public key
- \( Y \): public key
- \( P \): padding
- \( M \): mark
- \( E \): epoch
- \( k \): master key
Session tickets (similar to TLS)

Legend:
- **T**: ticket
- **P**: padding
- **M**: mark
- **E**: epoch
- **k**: master key

Client

\[ T_t \ || \ P \ || \ M \ || \ MAC_{k_t}(T_t \ || \ P \ || \ E) \]

\[ \text{Enc}_{k_t}(k_{t+1} \ || \ T_{t+1}) \]

**handshake complete**

\[ \text{Enc}_{k_t} (\text{Tor traffic}) \]

Server
How to distribute the 20-byte shared secret?
What does the shared secret look like?

- Base32 for easier distribution in meatspace.

- Example:

  Bridge scramblesuit 193.10.227.195:9002
  password=5TYVADJINHBB67PJSBPSWVR5I0742PVO
Active probing resistance is not enough!

- Tor could still be identified by its **flow properties**.
- E.g., 586-byte signature (512-byte cell + TLS + TCP + IP).
- Maybe even inter-arrival times.
- Our solution: A **unique** flow signature for every server!
One flow signature for every server

Unique random seed

PRNG

Frequency distributions
One flow signature for every server
One flow signature for every server
Packet length distribution

(a) Client-to-server.

(b) Server-to-client.
Inter-arrival time distribution

(c) Client-to-server.

(d) Server-to-client.
It’s not that easy, though

- **Strong defence** against traffic analysis doesn’t come for free!

- We ignored “total bytes transferred” and “traffic bursts” which are expensive to disguise.

- (Semi-)Expensive classifiers such as VNG++ are still problematic!
How (un)practical is it?

- Session tickets inexpensive and 1536-bit UniformDH OK.
- Pure Python implementation using PyCrypto reasonably fast.
- Packet length obfuscation and protocol header inexpensive.
- Inter-arrival obfuscation expensive!
- Would work in China, Syria, sometimes Iran.
Throughput

Based on transferring a 1,000,000-byte file:

<table>
<thead>
<tr>
<th></th>
<th>Tor</th>
<th>ScrambleSuit</th>
<th>ScrambleSuit-nodelay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodput</td>
<td>286 KB/s</td>
<td>148 KB/s</td>
<td>321 KB/s</td>
</tr>
<tr>
<td>Overhead</td>
<td>19.6%</td>
<td>52.1%</td>
<td>45.5%</td>
</tr>
</tbody>
</table>
Want to give it a try?

- Code and data: http://veri.nymity.ch/scramblesuit
- Developed ~2,600-lines prototype in Python.
- Will soon be deployed in pluggable transport Tor Browser Bundle.
Our first bridge is looking good
Contact

E-mail:
philipp.winter@kau.se

Project web site:
http://veri.nymity.ch/scramblesuit

Thanks to:
George Kadianakis
Harald Lampesberger
Stefan Lindskog
Michael Rogers
Internetfonden for research grant