TapDance: End-to-Middle Anticensorship without Flow Blocking

Eric Wustrow
Colleen M. Swanson
J. Alex Halderman
University of Michigan
End-to-Middle Proxies

**Telex**: Anticensorship in the Network Infrastructure

*Eric Wustrow, Scott Wolchok, Ian Goldberg, J. Alex Halderman*
*USENIX SEC 2011*

**Decoy Routing**: Toward Unblockable Internet Communication

*Josh Karlin, Daniel Ellard, Alden W. Jackson, Christine E. Jones, Greg Lauer, David P. Mankins, W. Timothy Strayer*
*FOCI 2011*

**Cirripede**: Circumvention Infrastructure using Router Redirection with Plausible Deniability

*Amir Houmansadr, Giang T. K. Nguyen, Matthew Caesar, Nikita Borisov*
*CCS 2011*
End-to-Middle Proxies

Client

E2M Proxy

Decoy Server
NotBlocked.com
End-to-Middle Proxies
End-to-Middle Proxies
**TapDance**: End-to-Middle Anticensorship without Flow Blocking
Passive Tap Challenges
Passive Tap Challenges
Passive Tap Challenge: Duplicate Data

Client → Censor → Decoy Server

K → (TLS Handshake) → K

<Encrypted HTTP Request>

<Encrypted E2M Response>

<Encrypted HTTP Response>
Incomplete HTTP request example

GET / HTTP/1.1\r\nHost: www.site.com\r\nX-Ignore: AAAAAAAAAAAAAA...\r\n
Incomplete HTTP Request

Client → Censor → TapDance Proxy → Decoy Server

(K) "(TLS Handshake)" (K)

<Encrypted Incomplete HTTP Request>

<Encrypted E2M Response>
Passive Tap Challenge: Inconsistent TCP

Client → Censor → TapDance Proxy → Decoy Server

(K) → (TLS Handshake) → (K)

<Encrypted Incomplete HTTP Request>

<Encrypted E2M Response>

TCP ACK (<Encrypted E2M Response>)
Passive Tap Challenge: Inconsistent TCP

Client → Censor → TapDance Proxy → Decoy Server

(K) ← (TLS Handshake) → (K)

<Encrypted Incomplete HTTP Request>

<Encrypted E2M Response>

TCP ACK (<Encrypted E2M Response>)

<Encrypted E2M Request>
<table>
<thead>
<tr>
<th>Tagging Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telex:</td>
<td>28-byte TLS \textit{ClientRandom}</td>
</tr>
<tr>
<td>Decoy Routing:</td>
<td>28-byte TLS \textit{ClientRandom}</td>
</tr>
<tr>
<td>Cirripede:</td>
<td>Successive (3-byte) TCP ISNs</td>
</tr>
<tr>
<td>TapDance:</td>
<td>Arbitrary-length TLS \textit{Ciphertext}</td>
</tr>
</tbody>
</table>
Chosen-Ciphertext Covert Channel

Alice (Client)  Enc  Encrypted Channel  Dec  Bob (Server)

Eve (E2M Proxy)
Stream cipher ciphertext channel

AES

$P_0$

47 45 54 20 2f ..

$C_0$

64 5e 59 48 d4 ..

AES

$P_1$

00 00 00 00 00 ..

$C_1$

26 5e df 61 22 ..
Stream cipher ciphertext channel

\[ \text{AES}_{P_0}(47 45 54 20 2f \ldots) = 64 5e 59 48 d4 \ldots \]

\[ \text{AES}_{P_1}(01 00 00 00 00 \ldots) = 27 5e df 61 22 \ldots \]
Stream cipher ciphertext channel

0

AES

P_0

47 45 54 20 2f ..

C_0

64 5e 59 48 d4 ..

1

AES

P_1

27 5c dc 65 27 ..

C_1

01 02 03 04 05 ..
Stream cipher ciphertext channel

AES

\[ P_0 \rightarrow C_0 \}

\[ P_1 \rightarrow C_1 \}

\[ 47 \ 45 \ 54 \ 20 \ 2f \ .. \]

\[ 07 \ 0c \ 05 \ 07 \ .. \]

\[ 64 \ 5e \ 59 \ 48 \ d4 \ .. \]

\[ c1 \ 92 \ 43 \ 64 \ f5 \ .. \]
Incomplete HTTP request example

GET / HTTP/1.1

Host: www.site.com

X-Ignore: u]DhsYGxVxEvuZEHES...
Incomplete HTTP request example

```
x1e\x91\xb2\xce\x94\x8a\x6b\x3c\x78\x8c\x6f\x03
\x5e\xef\x97\x34\xf1\x2e\xc6\xe6\x7f\x10\xc8\x46
\xf9\x25\x6a\x0c\xff\x6d\x38 ... \x70\xd7\x2c\x63 ...
```

Decrypt

Shared Secret: ; Client random: ...
TapDance Protocol Overview

Client → Censor → TapDance Proxy → Decoy Server

K → (TLS Handshake) → K

<Encrypted Incomplete HTTP Request>

Tag

<Encrypted TapDance Response>

<Encrypted Proxy Request (blocked.com)>

<Encrypted HTTP Response (blocked.com HTML)>
TapDance Active Attack

Client → Censor → TapDance Proxy → Decoy Server

(TLS Handshake)

$K \rightarrow \rightarrow K$

<Encrypted Incomplete HTTP Request>

<Encrypted TapDance Response> (seq=Y, ack=X)

TCP stale ACK (seq=X-1, ack=Y)

TCP ACK (seq=Y, ack=X)
Hosts that allow Incomplete Requests

![Graph showing fraction of hosts that timeout after X seconds against timeout (seconds). The graph compares Alexa top 1 million and IPv4 sample.]
# Previous work comparison

<table>
<thead>
<tr>
<th></th>
<th>Telex</th>
<th>Cirripede</th>
<th>Decoy Routing</th>
<th>TapDance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steganographic channel</td>
<td>ClientRandom</td>
<td>TCP ISN</td>
<td>ClientRandom</td>
<td>TLS Ciphertext</td>
</tr>
<tr>
<td>No Inline blocking</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
</tr>
<tr>
<td>Asymmetric flows</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Replay attack defense</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Traffic analysis defense</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Future Work

• Real world deployment
• “Single-packet” (stateless) TapDance
• Traffic fingerprinting defense
• Active defense
Conclusion

- **TapDance** removes **inline blocking** requirement for End-to-Middle proxies, facilitating deployment
- Continues to function with **asymmetric traffic**
- Tradeoff of **active attack vulnerability** versus deployability