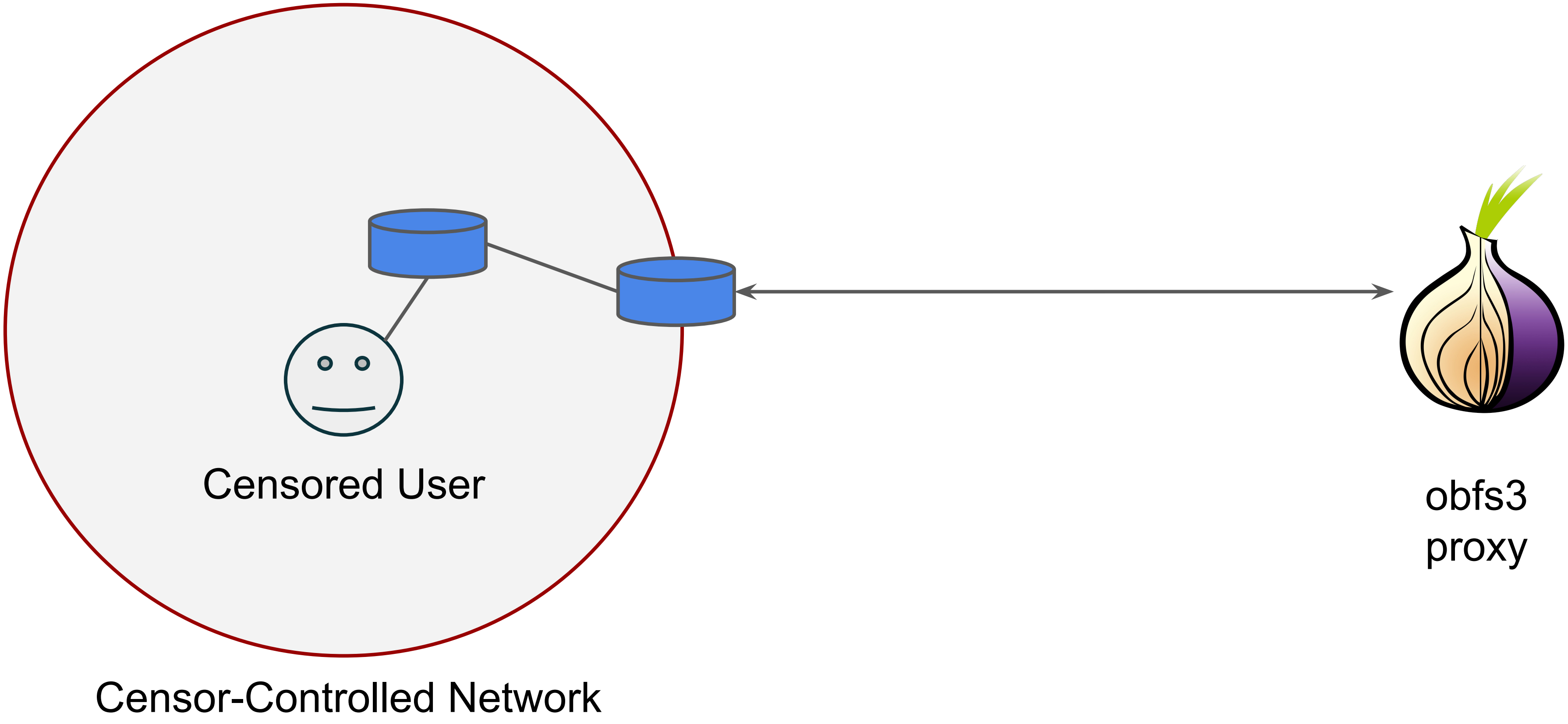
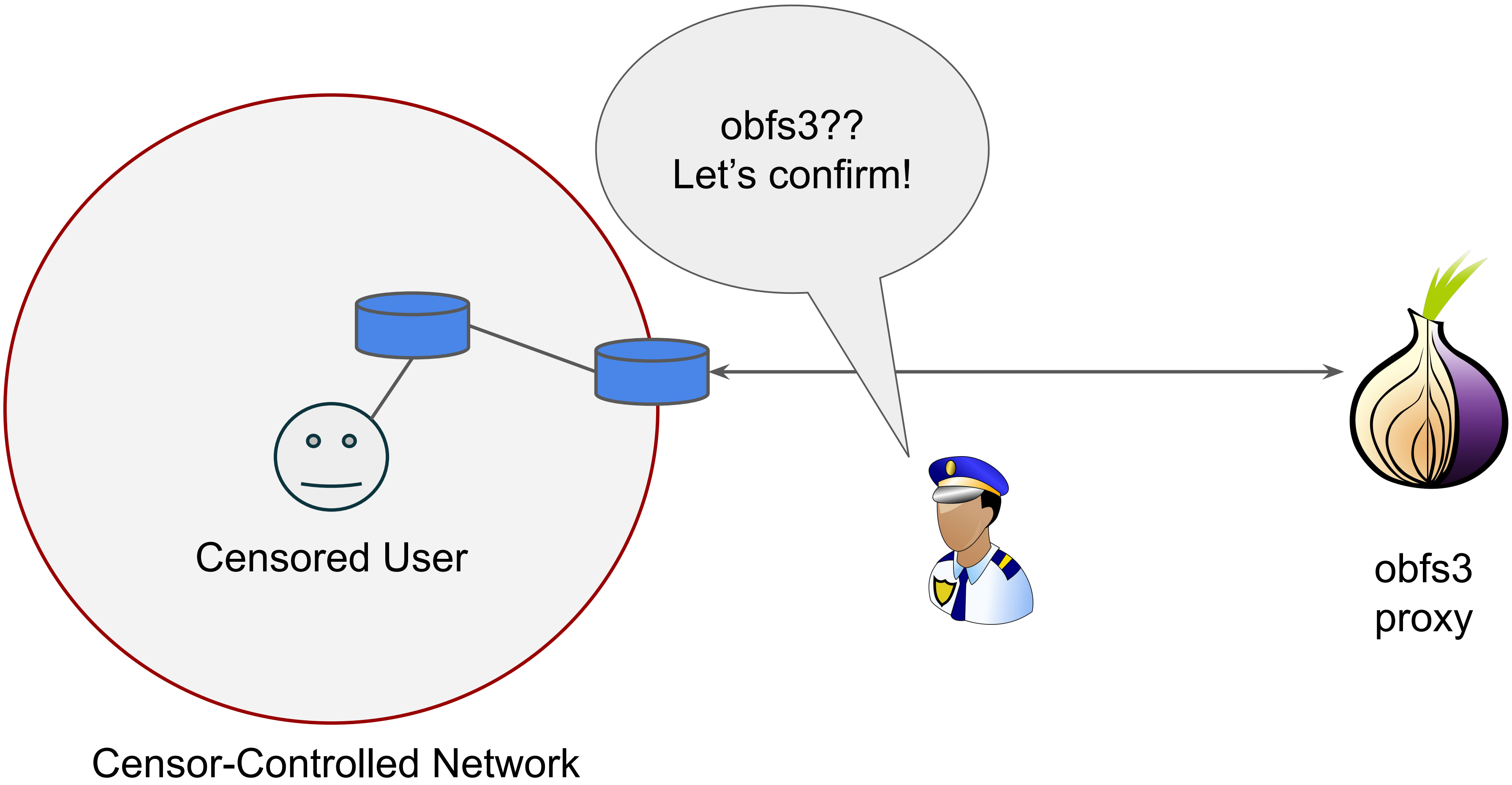
Detecting Probe-resistant Proxies

Sergey Frolov, Jack Wampler, Eric Wustrow
University of Colorado Boulder

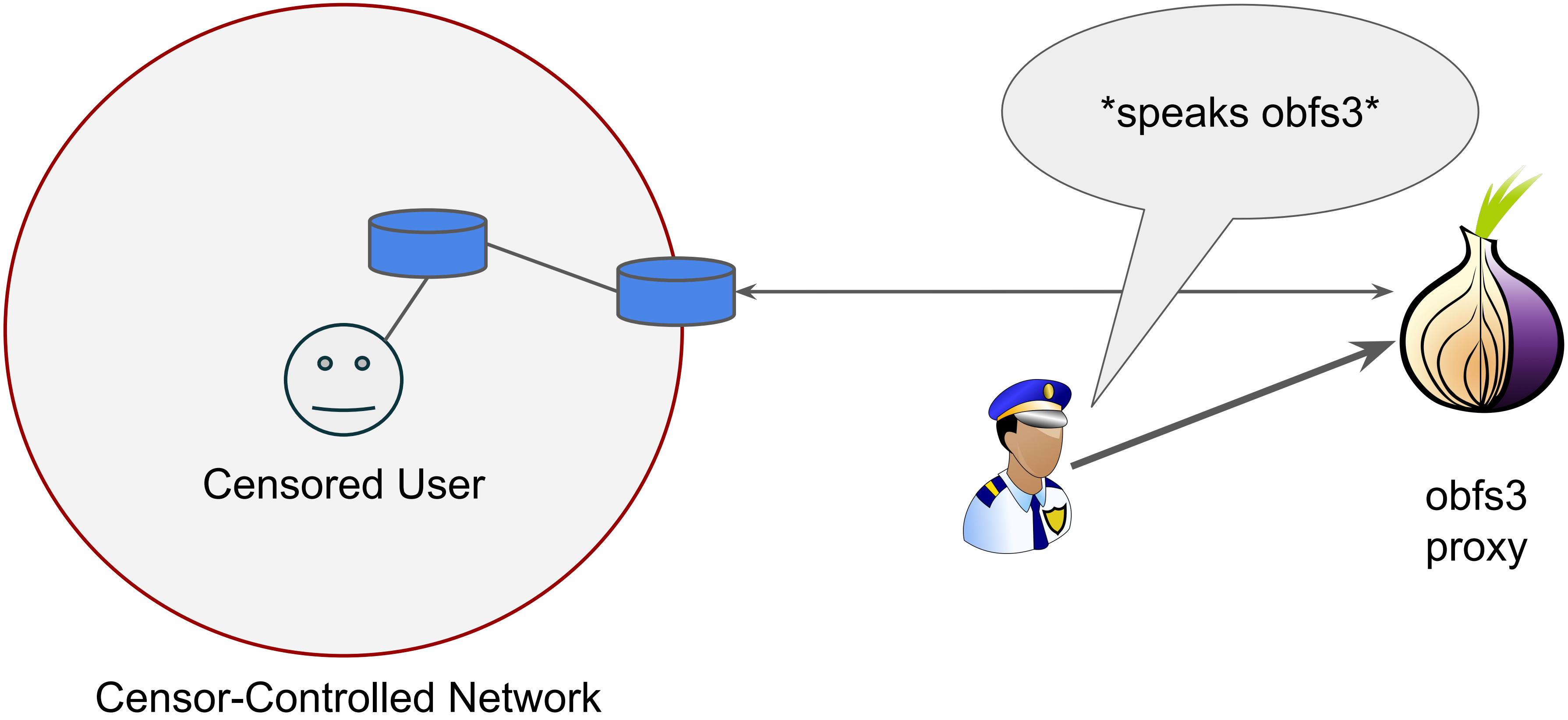
Proxies



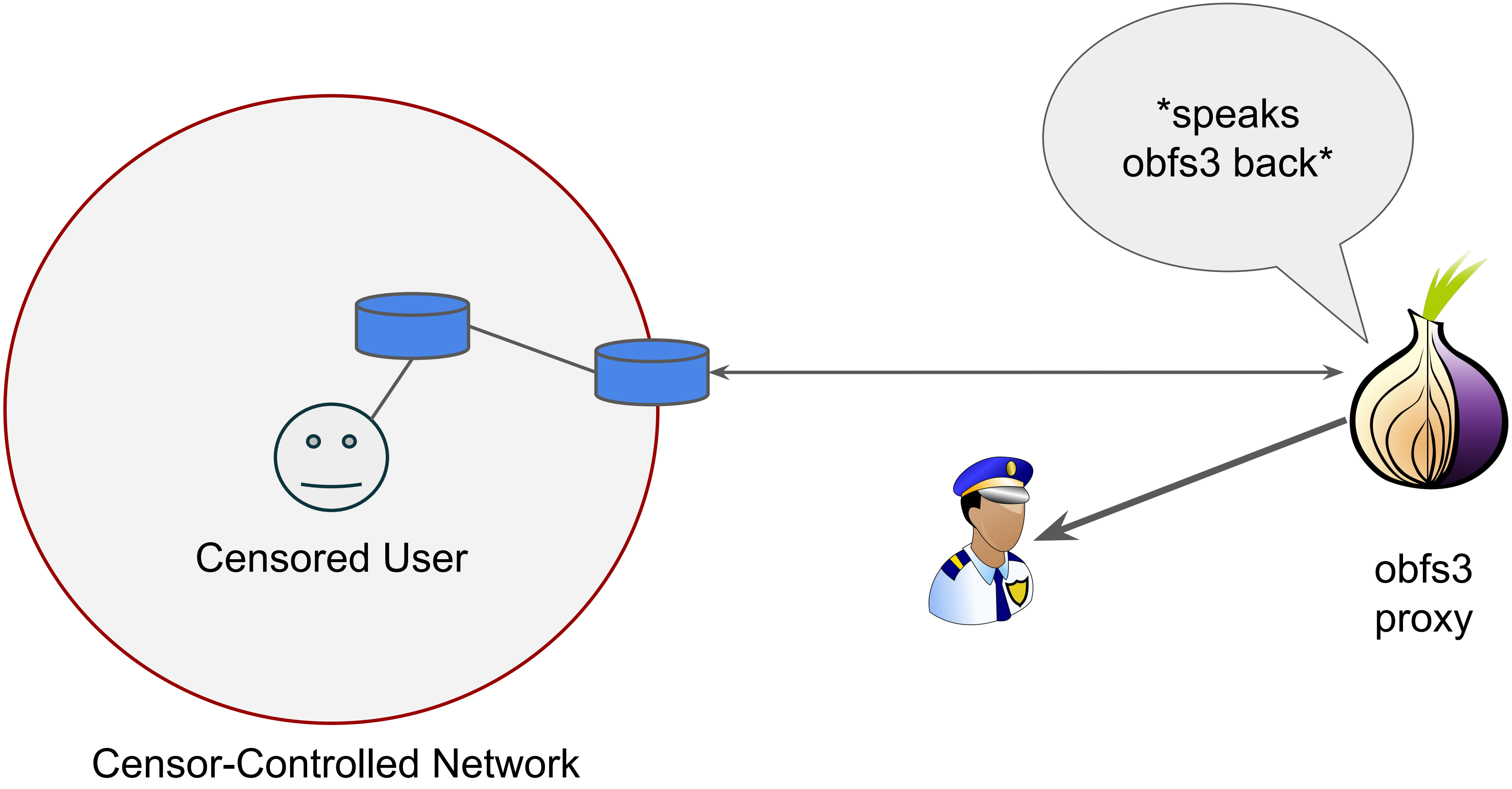
Active Probing



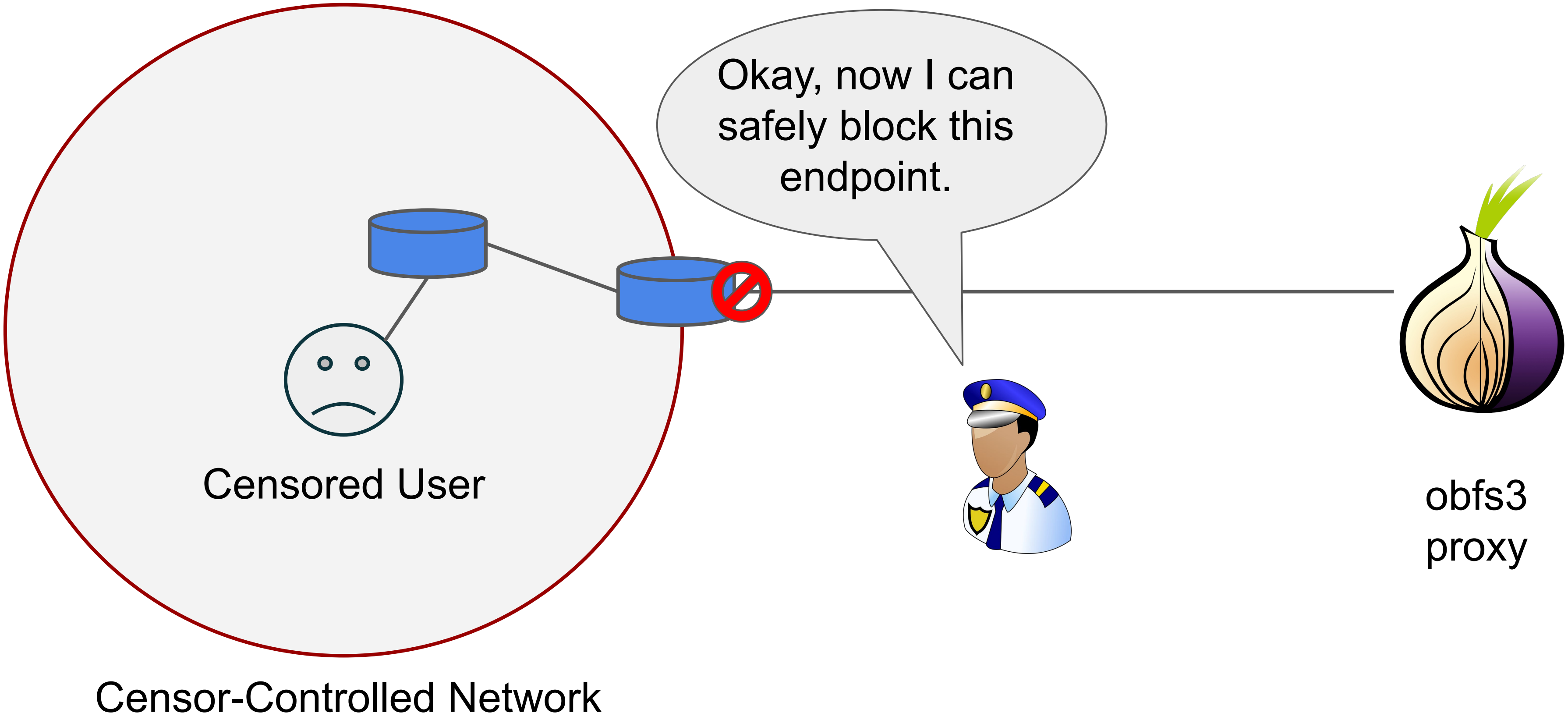
Active Probing



Active Probing



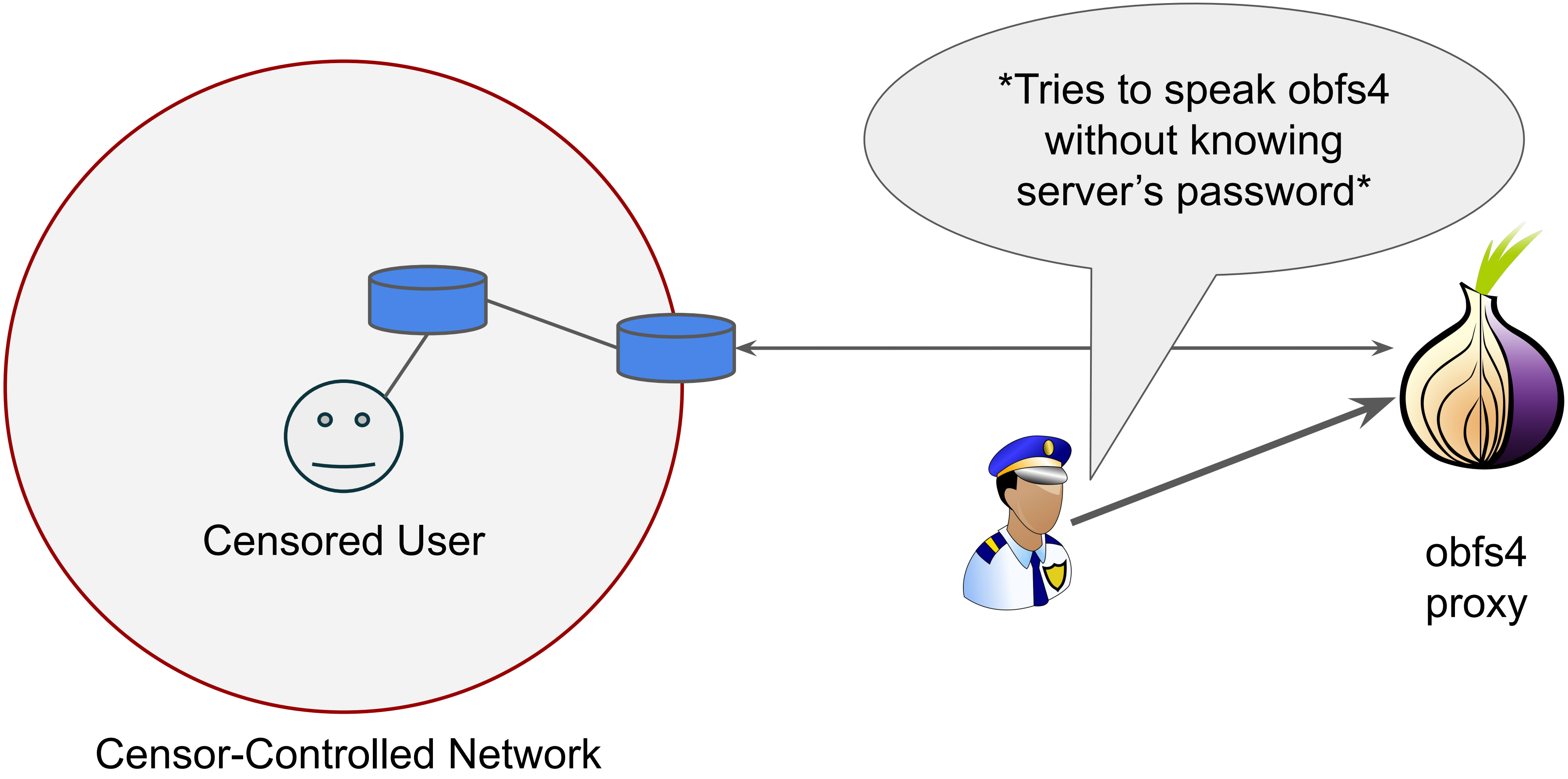
Active Probing


Okay, now I can safely block this endpoint.
Censored User
obfs3 proxy
Censor-Controlled Network

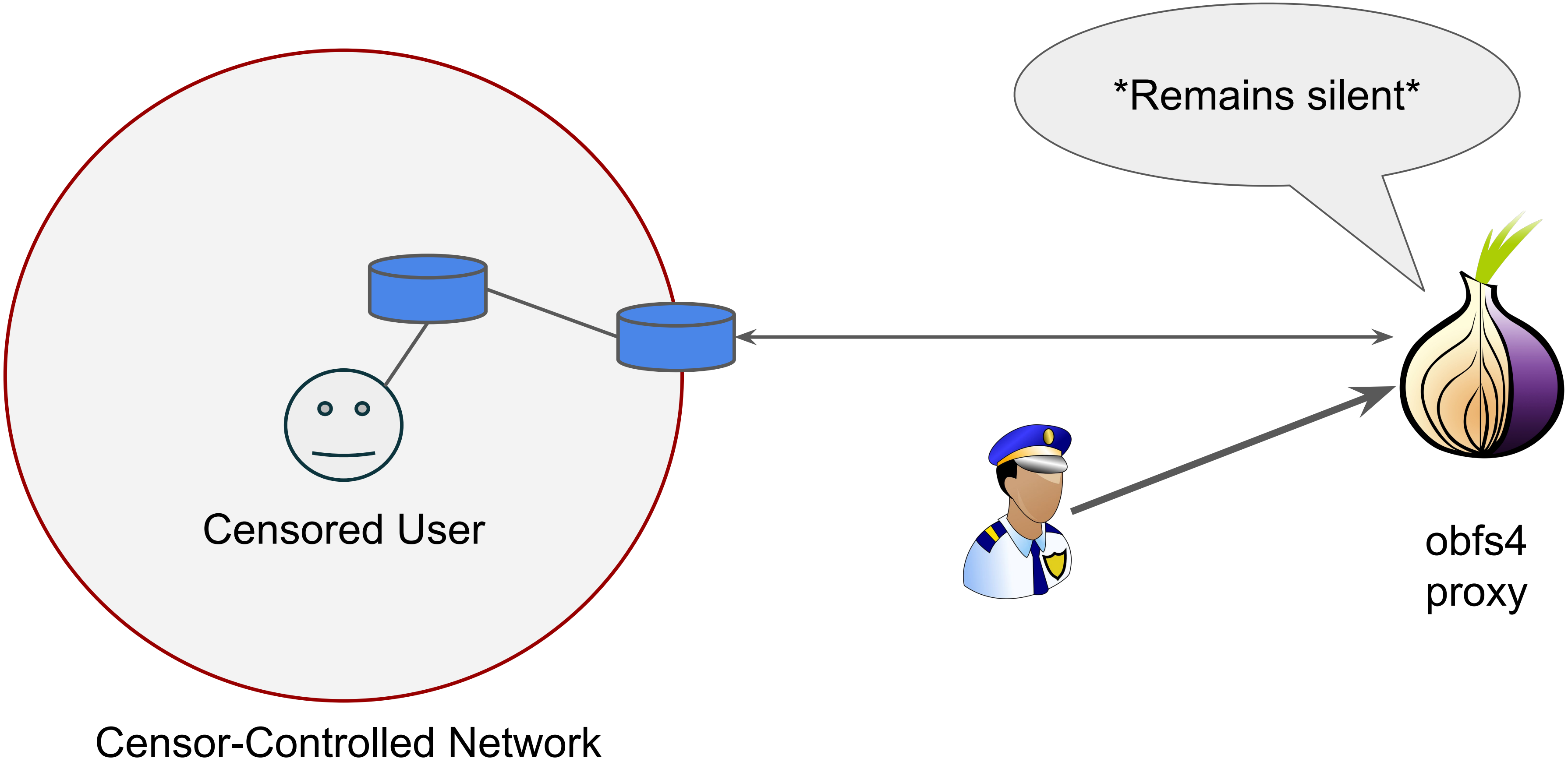
Thwarting Active Probing

- Probe-Resistant proxies
  - Require knowledge of **shared secret** to use
  - Don’t know secret? Server remains **silent**

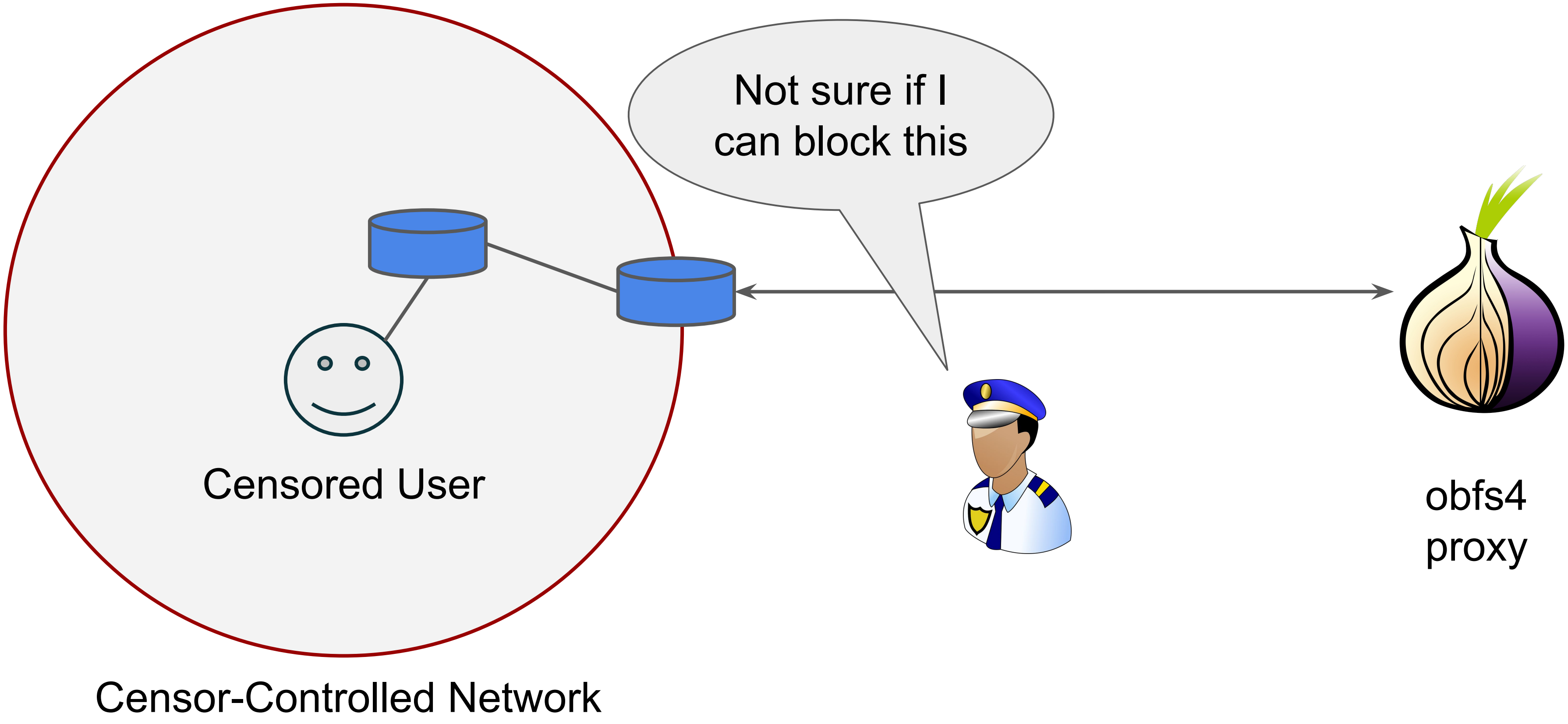
Thwarting Active Probing



Thwarting Active Probing



Thwarting Active Probing



Probing Probe-Resistant proxies

Are these proxies actually probe-resistant in practice?

- How **common** is the behavior of proxies to never respond to HTTP, TLS, ...any protocol?
    - If not common, censor can block it.

Probing Probe-Resistant proxies

We need a source of TCP endpoints on the internet to compare their responses with Probe-Resistant proxies' responses. We have 2 datasets:

ZMap Dataset
785k endpoints

Tap Dataset
433k endpoints

Probing Probe-Resistant proxies

We used the following probes:

1. HTTP
2. TLS ClientHello
3. Modbus
4. S7
5. Random bytes (23B - 17KB)
6. Empty probe
7. DNS zone Transfer
8. STUN

Probing Probe-Resistant proxies

For each probe we record *3-tuple result*:

- Time to close
- Type of close (FIN, RST or TIMEOUT)
- Size of response data
    - Probe-resistant proxies *never* respond!

Endpoints that respond with data

Probe	Tap dataset
TLS	87.8%
HTTP	64.6%
DNS-AXFR	58.8%
S7	56.9%
STUN	52.5%
Modbus	51.4%
Empty	8.4%
Any	**94.0%**

Response *alone* can distinguish 94% of endpoints in the realistic Tap dataset from proxies.

Endpoints that respond with data

Probe	Tap dataset	ZMap dataset
TLS	87.8%	0.90%
HTTP	64.6%	0.95%
DNS-AXFR	58.8%	0.67%
S7	56.9%	0.66%
STUN	52.5%	0.56%
Modbus	51.4%	0.54%
Empty	8.4%	0.23%
Any	**94.0%**	**1.16%**

Very few “legitimate” services
(lots of firewalls/honeypots)

Probing Probe-Resistant proxies

How do our probe-resistant proxies respond to those probes?

We examine:

- obfs4
- ObfuscatedSSH
- Lampshade
- MTProto Proxy
- Shadowsocks-Outline
- Shadowsocks-Python

Probing ObfuscatedSSH

How else can we distinguish proxies from remaining 6%?

Probe	Close Time (s)	Close Type
Modbus	30.237	FIN
S7	30.236	FIN
Random 23	30.238	FIN
Empty probe	30.238	FIN

Probe	Close Time (s)	Close Type
HTTP GET	0.250	RST
TLS ClientHello	0.240	RST
Random 25, 47, 51, 7KB, 17KB	0.237 - 0.251	RST
DNS AXFR	0.242	RST
STUN	0.236	RST

Proxy server code

```
clientConn := listener.Accept()
```

Proxy server code

```
clientConn := listener.Accept()

clientConn.SetDeadline(in30Seconds)
```

Proxy server code

```
clientConn := listener.Accept()

clientConn.SetDeadline(in30Seconds)

buffer := make([]byte, 50)
```

Proxy server code

```
clientConn := listener.Accept()

clientConn.SetDeadline(in30Seconds)

buffer := make([]byte, 50)
error := io.ReadFull(clientConn, buffer)
if error != nil { // didn’t get 50 bytes in 30s
    clientConn.Close()
    return
}
```

Proxy server code

```
clientConn := listener.Accept()

clientConn.SetDeadline(in30Seconds)

buffer := make([]byte, 50)
error := io.ReadFull(clientConn, buffer)
if error != nil { // didn’t get 50 bytes in 30s
    clientConn.Close()
    return
}

if !checkCredentials(buffer) {
    clientConn.Close()
    return
}
// do the proxying here
```

Close Thresholds

Probe Size	Response Size	Close Time	Close Type
49 bytes or fewer	0	30s	FIN
50 bytes	0	Right away	FIN
51 bytes or more	0	Right away	RST

Can probe-resistant proxies be distinguished from other servers due to such thresholds?

Investigating Close Thresholds

- Built a threshold scanner to **binary search** for close thresholds
  - Send random data of different lengths
  - Scanned Tap/ZMap endpoints to compare with probe-resistant proxies
  - Check for “stability”

Proxies’ thresholds

Proxy	FIN Threshold	RST Threshold
ObfuscatedSSH	24 B	25 B
Shadowsocks-Python	50 B	-
Shadowsocks-Outline	50 B	51 B
Lampshade	256 B	257 B
obfs4	8 KB - 16 KB	next mod 1448
MTProto	-	-

Investigating Close Thresholds

	Tap Dataset	ZMap Dataset
Endpoints	433k	779k
“Stable” thresholds	**144k (33.5%)**	**116k (15%)**

Investigating Close Thresholds

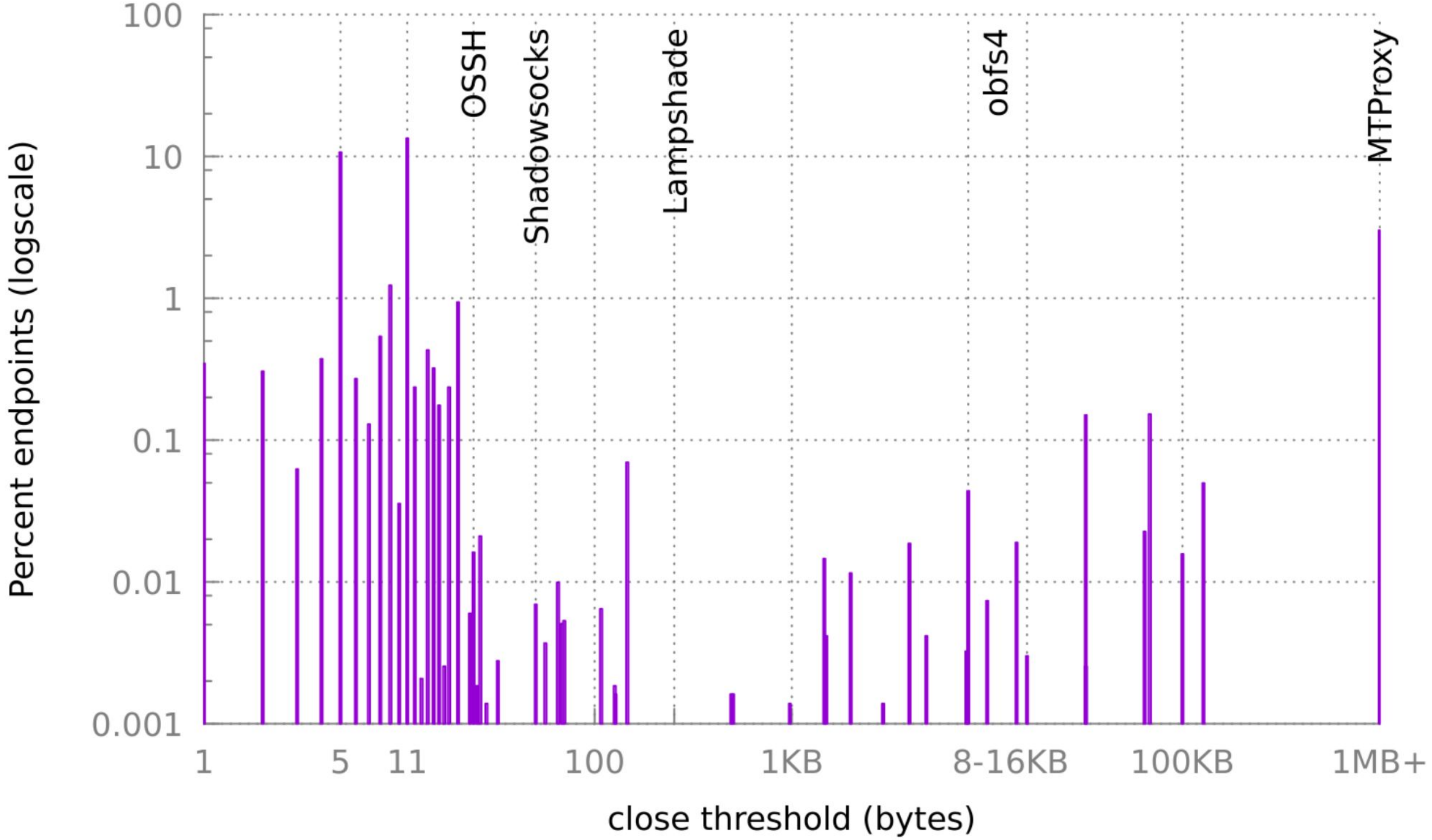
	Tap Dataset	ZMap Dataset
Endpoints	433k	779k
“Stable” thresholds	144k (33.5%)	116k (15%)

Why so few stable close thresholds?

Sent data response	**257k (59.5%)**	5k (0.7%)
Error	3k (0.8%)	**568k (73%)**
“Unstable” thresholds	27k (6.2%)	88k (11.3%)

Tap Endpoints' Stable Thresholds

5, 11 and no threshold are the most common.

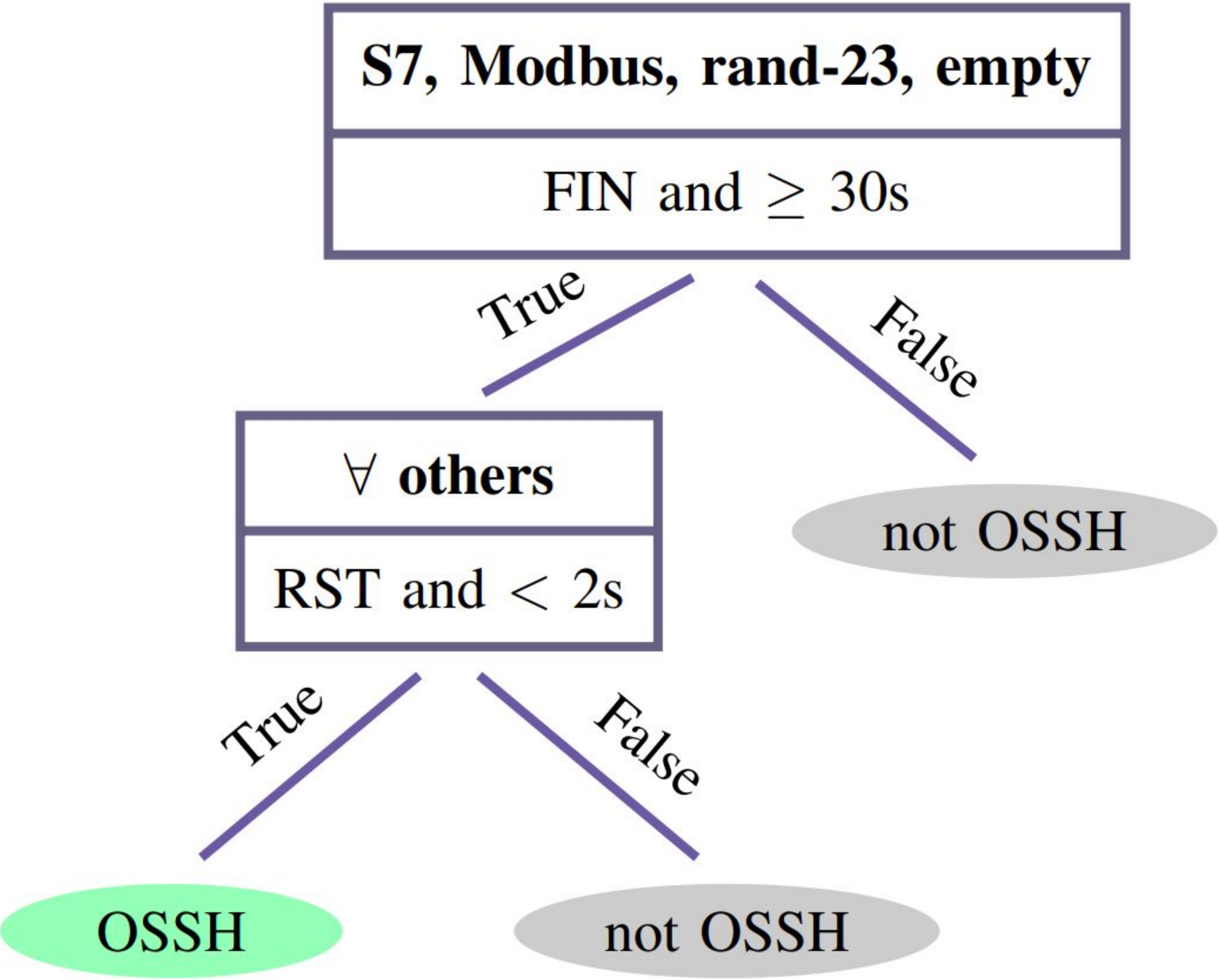


Decision Trees

We built manual decision trees to detect Probe-Resistant proxies based on their responses to our probes.

We also evaluated automatic decision trees, but they seemed less practical (see Appendix).

Manual ObfuscatedSSH decision tree



S7, Modbus, rand-23, empty

FIN and $\geq$ 30s

True

False

not OSSH

∀ others

RST and $<$ 2s

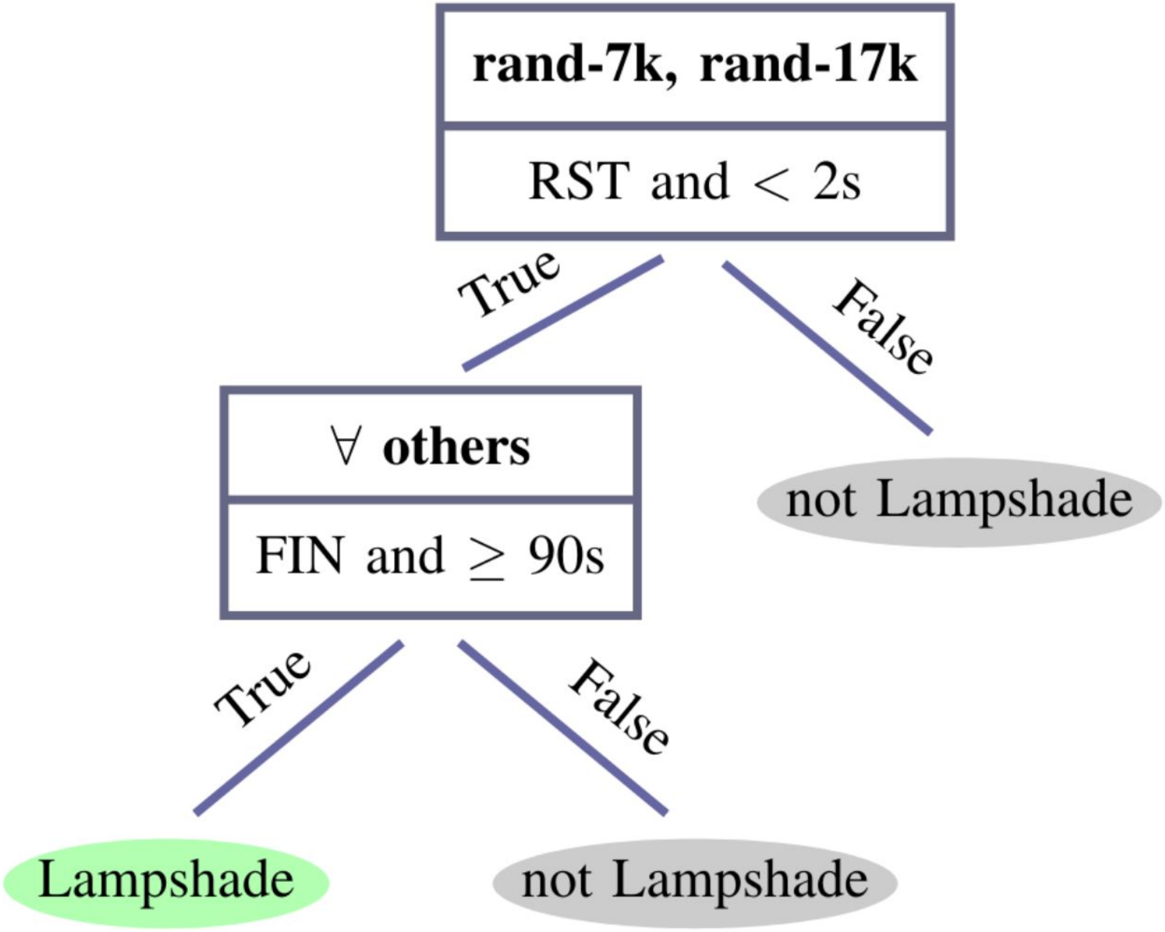
True

False

OSSH

not OSSH

Manual Lampshade Decision Tree



rand-7k, rand-17k

RST and < 2s

True

False

not Lampshade

∀ others

FIN and ≥ 90s

True

False

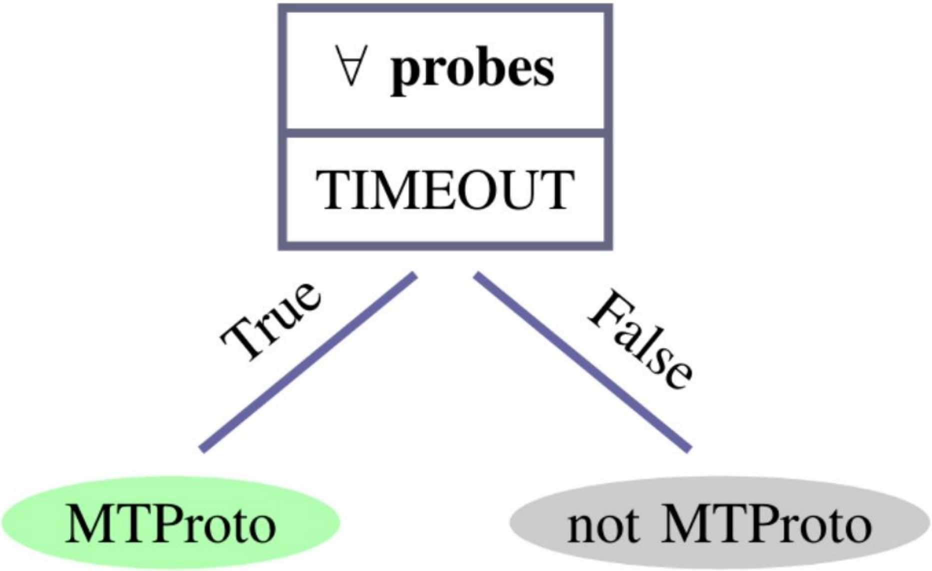
Lampshade

not Lampshade

Decision tree results

Proxy	Decision Tree Labeled	
	Tap	ZMap
Lampshade	0	1
ObfuscatedSSH	8	0
obfs4	2	0
Shadowsocks-Python	0	8
Shadowsocks-Outline	0	7
MTProto	3144	296

Manual MTProto decision tree



Defense Strategies

- Recommended: never respond, never close connection
  - 0.56% of Tap dataset
- Randomizing parameters, such as timeout, on a per-server basis increases the overall size of “Anonymity Set” for your transport.
- Stable thresholds are a *fingerprint*
  - To fix don’t close immediately after handshake fails and keep draining the buffer until the timeout

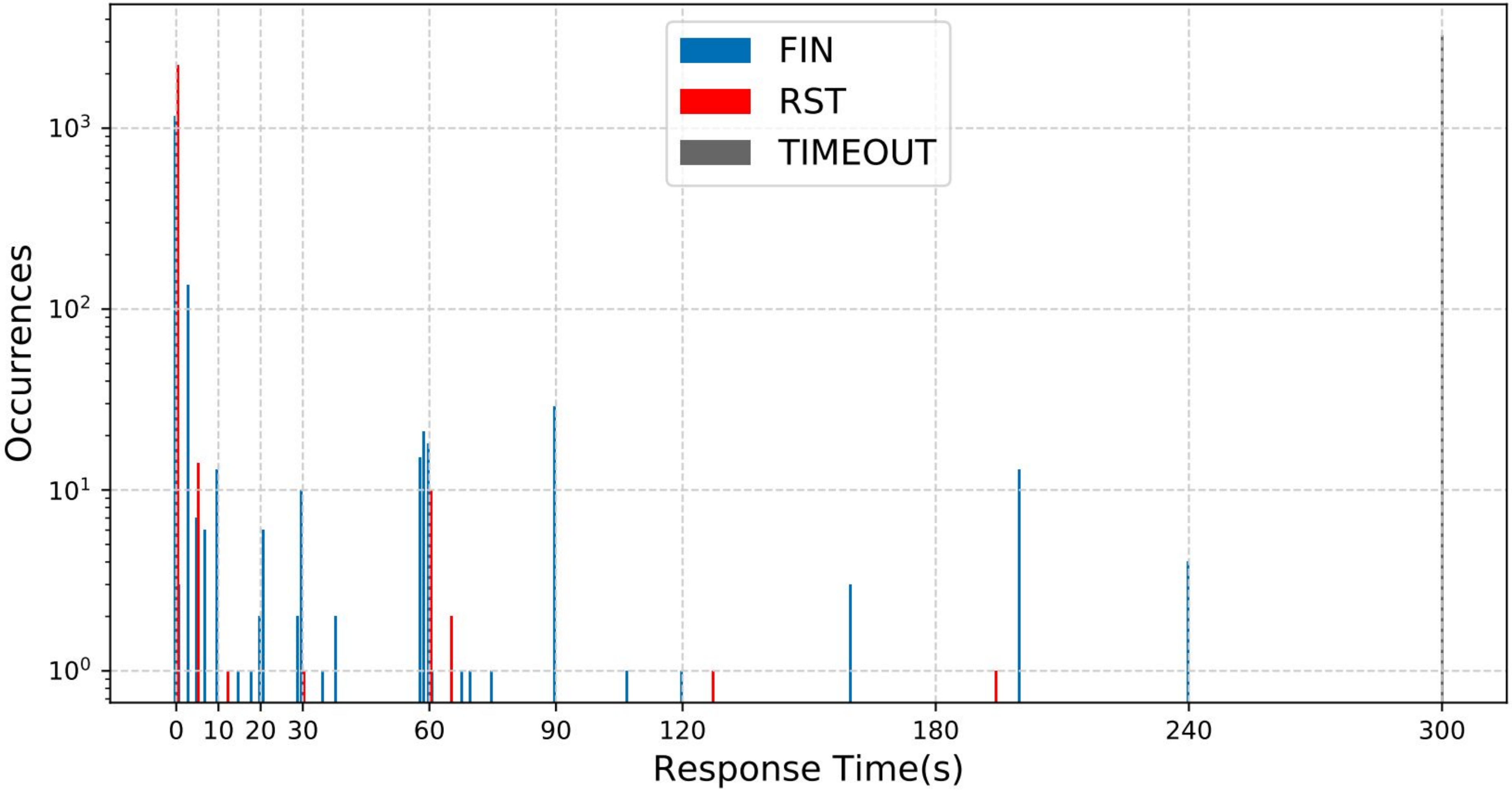
Responsible Disclosure

We disclosed the presence of unique close thresholds to the devs, and as a result, it was removed from:

- OSSH on May 13, 2019
- obfs4 on June 21, 2019 (version 0.0.11)
- SS-Outline on September 4, 2019 (version 1.0.7)
- Lampshade on October 31, 2019

Timeouts still have to be chosen with care.

Probe-indifferent Server Timeouts (Tap)



But note: popular values might be limited to specific applications

Conclusions

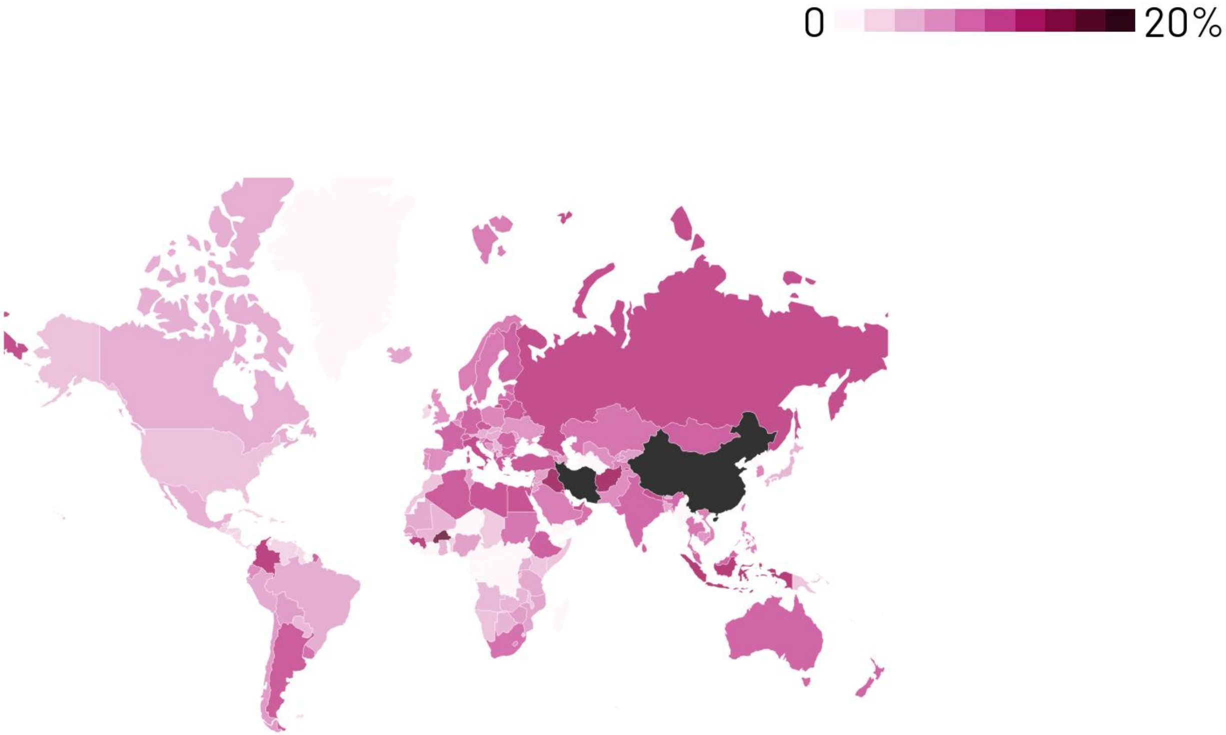
- Probe-resistant proxies aren’t (or weren’t!)
  - Never responding with data is uncommon on the Internet
  - Connection timeouts and thresholds can be used to fingerprint server applications
- Notified proxy developers
  - Removed thresholds
  - But choosing timeouts still tricky
- Long-term: investigate alternative proxy protocols
  - e.g. Domain Fronting, Refraction, HTTPS-proxy

FIN

Thank you for attention!

Backup

Internet Censorship



Mean percentage of domains from Satellite input list blocked per country.

Source: https://censoredplanet.org/data/visualizations

https://gfw.report

- “**How China Detects and Blocks Shadowsocks**” describes evidence of a similar active probing attack occuring in China in 2019.

Removing Close Threshold

How to fix this behavior?

Probe Size	Response Size	Close Time	Close Type
49 bytes or fewer	0	30 sec	FIN
50 bytes	0	Right away	FIN
51 bytes or more	0	Right away	RST

Removing Close Threshold

```
clientConn := listener.Accept()

clientConn.SetDeadline(in30Seconds)

buffer := make([]byte, 50)
error := io.ReadFull(clientConn, buffer)
if error != nil { // didn’t get 50 bytes in 30s
    clientConn.Close()
    return
}

if !checkCredentials(buffer) {
    clientConn.Close()
    return
}
// do the proxying here
```

Removing Close Threshold

```
clientConn := listener.Accept()

clientConn.SetDeadline(in30Seconds)

buffer := make([]byte, 50)
error := io.ReadFull(clientConn, buffer)
if error != nil { // didn’t get 50 bytes in 30s
    clientConn.Close()
    return
}

if !checkCredentials(buffer) {
    io.Copy(ioutil.Discard, clientConn)
    clientConn.Close()
    return
}
```

Removing Close Threshold

Probe Size	Response Size	Close Time	Close Type
49 bytes or fewer	0	30 sec	FIN
50 bytes	0	30 sec	FIN
51 bytes or more	0	30 sec	FIN