Pluggable Transports

Practical Methods for Combating Deep Packet Inspection

David Oliver
Guardian Project
IETF103

TL / DR

- Pluggable Transports (PT) are a defense against Internet censorship and surveillance.
- A PT obfuscates the address or contents of network flows, protecting from intermediaries who deploy Deep Packet Inspection (DPI).
- A PT is “pluggable” if written to a semi-formal specification currently being developed in the Internet Freedom community
- A PT needs to be “pluggable” because the half-life of a given obfuscation technique is short and rapid update is necessary.
- Obfuscation requires partnership - a “client” and a “bridge” / “gateway”
- PTs are (currently) written into VPN services, local (client) proxies, or directly in apps.
- “Pluggable“ implemented at developer level, not user level

Surveillance: Why should we care?

- A gentle reminder that, over time, the Internet has provided an increasingly stark lesson that surveillance runs at cross-purposes to individual privacy.
- Privacy is among the core concepts enabling Human Rights.
- NOTE: Guardian Project focuses on mobile or embedded devices using the mobile/wireless networks
    - Generally, these devices lack, or are underserved by. privacy technologies easily available on the desktop

Deep Packet Inspection (DPI)

- Once upon a time, IP served to route content packets around the network.
- But IP routing data got weaponized - collected, measured, correlated and aggregated for uses other than routing.
- The newest weapon is traffic analysis by Deep Packet Inspection (DPI)
  - Ever more powerful routing equipment now allows inspection of the contents of every packet.
  - Encryption helps, but encrypted content nonetheless adheres to recognizable patterns.
    - E.g. “this packet starts a Tor network connection”
- Now, by “reading all your mail”, the Internet postman has a whole new range of data to collect, measure, correlate, and aggregate.
  - Inherently, this is an entirely new level of surveillance

Can we defend against DPI?

- Yes - through obfuscation
  - Through transformation of a packet's data into a "benign" form that does not arouse suspicion.
  - Through obscuring a transaction's intended service endpoint.
    i. In both cases, an "un-transformation" is required before the transaction is serviced.
- But what doesn't arouse suspicion today might arouse suspicion tomorrow
  - Obfuscation is a moving target
  - Application developers need a way to keep up with improving obfuscation technology
  - Those developing obfuscation technology need a way to deliver it more rapidly
- A technique for (relatively) rapid update was posited in 2012: PLUGGABLE TRANSPORTS
  - Initial work by the Tor Project

How does obfuscation technology work? (1)

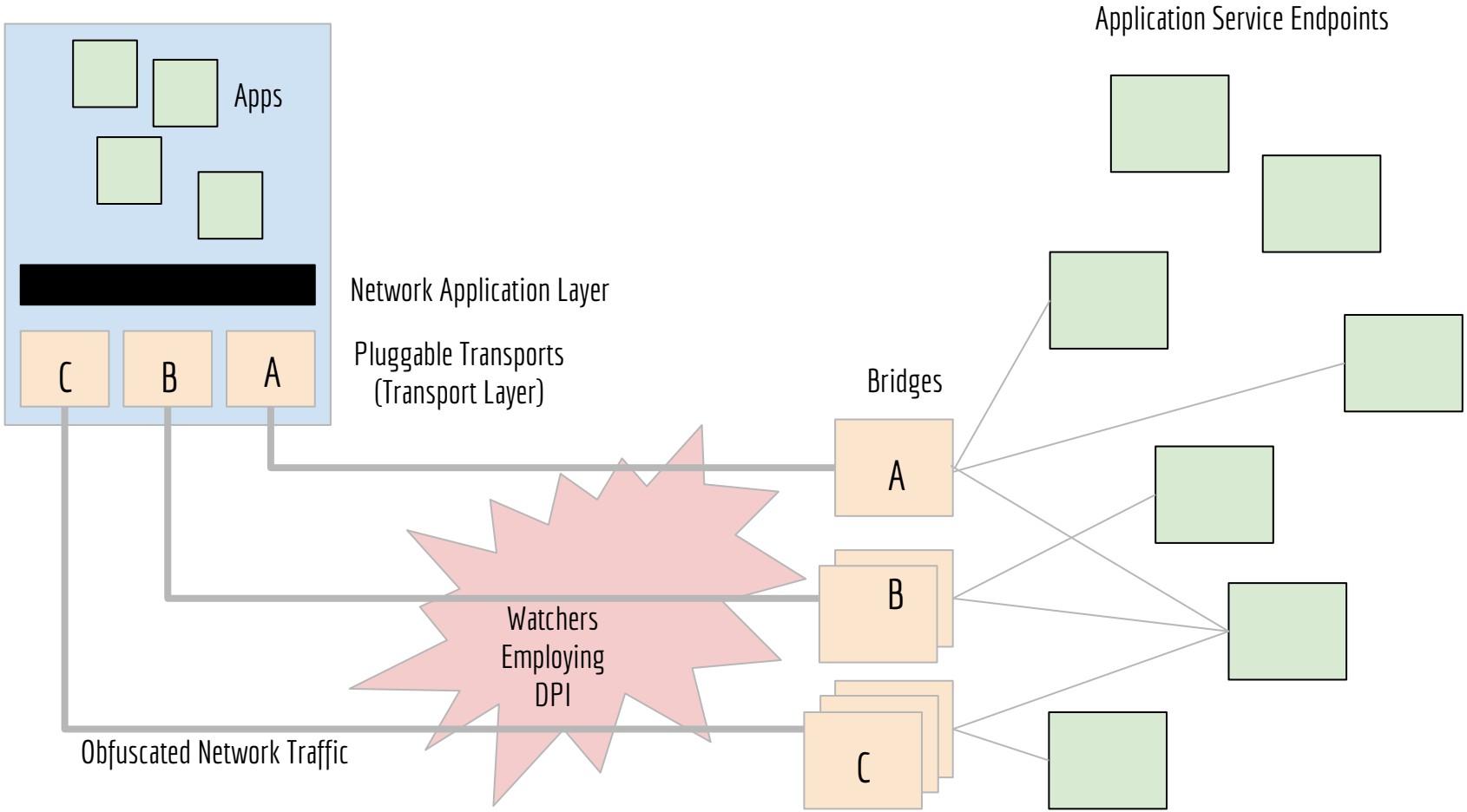
- PT operates at the network Transport layer
  - Allows the application to believe it is merely reading/writing bytes on its communication endpoint
  - Independently, transform those bytes as necessary
    i. Pluggable Transports leverage this independence to insert obfuscation options
- Techniques of obfuscation:
  - Fronting - hide the actual address-of-service
  - Scrambling - mash up the byte stream until recognizable patterns are removed
  - Shape-Shifting - make byte stream look like another byte stream
    i. E.g., make Tor traffic look like music streaming
       1. ...because music streaming is “benign” (not interesting to surveil)

How does obfuscation technology work? (2)

- Obfuscation requires cooperation between a client (user device) and a network intermediary
  a. A “bridge” that understands the form of obfuscation being used and “un-transforms” (un-obfuscates) it.
  b. Application service endpoints do not see obfuscated traffic.
- The bridging approach is common to Tor and VPNs
  a. But these bridges are targets for censorship so their addresses are often...mercurial
  b. Managing configuration of bridges is a crucial problem for Tor and some VPNs
     i. ...and made more difficult with PT (bridge-per-obfuscation-type).
- Under investigation are mechanisms for “crowd-sourcing” bridge support (using certain IoT devices, e.g.)

PT Client-side Implementation (current)

- VPN
  - Mobile operating system defines special interface for VPN technology
    - VPNService (Android), Network Extension (iOS)
    - Allows the VPN to insert itself at the TCP Transport Layer, independent of individual applications
  - Pluggability implemented within this framework, transparent to user except for configuration
- Tor
  - iOS
    - Desktop Tor binary (with wrappers) bundled as library with each application
    - Pluggability implemented by Desktop Tor
  - Android
    - Desktop Tor binary (with wrappers) implemented by single application (“Orbot”)
    - Applications are written to connect to Orbot as a SOCKS proxy or as a VPN (VpnService)


Application Service Endpoints
Apps
Network Application Layer
Pluggable Transports
(Transport Layer)
C
B
A
Bridges
A
B
C
Watchers
Employing
DPI
Obfuscated Network Traffic

Types of obfuscation: Currently-deployed PTs

- ScrambleSuit
  - Description: Is a transport that protects against follow-up probing attacks and is also capable of changing its network fingerprint (packet length distribution, inter-arrival times, etc.).
- obfs4
  - Description: Is a transport with the same features as ScrambleSuit but utilizing Dan Bernstein's elligator2 technique for public key obfuscation, and the ntor protocol for one-way authentication. This results in a faster protocol. Replaces 'obfs1', 'obfs2', 'obfs3'.
- meek
  - Description: Is a transport that uses HTTP for carrying bytes and TLS for obfuscation. Traffic is relayed through a third-party server (Google App Engine). It uses a trick ('domain fronting') to talk to the third party so that it looks like it is talking to an unblocked server.
- Format-Transforming Encryption (FTE)
  - Description: Is a transport that transforms Tor traffic to arbitrary formats using their language descriptions.

Types of obfuscation: Undeployed PTs

- StegoTorus
  - Description: is an Obfsproxy fork that extends it to a) split Tor streams across multiple connections to avoid packet size signatures, and b) embed the traffic flows in traces that look like HTML, JavasCript, or PDF. See its git repository.
- SkypeMorph
  - Description: It transforms Tor traffic flows so they look like Skype Video. See its source code and design paper.
- Dust / Dust2
  - Description: It aims to provide a packet-based (rather than connection-based) DPI-resistant protocol. See its git repository.
- Snowflake
  - Description: uses WebRTC, inspired by Flashproxy (“flash” = “quick” / “short lived”)

Future PT work at Guardian Project

- Android
  - Pluggable Transports as Android System Service
- iOS
  - “iObfs”: iOS Network Extension implementing Pluggable Transports
- Bridge “crowdsourcing”
  - “flash” bridges on the Internet of Things
- Usability: bridge configuration
  - easier bridge acquisition and automated configuration via Mote w/o Domain Fronting

PT Standards

- Evolution:
  - PT “v0.1” (2012-14) - Tor only (un-pluggable)
  - PT “v1.0” (2014-16) - mobile Tor (Orbot) (quasi-pluggable) + industry interest
  - PT “v2.0” (2017-18) - Tor and industry providers (lightly-pluggable)
    - PT “v2.0” Specification
    - Still held back by incompatibility of mobile OS programming environments & process models
- Opportunities:
  - Standardize new mechanism for delivering obfuscation services at a lower level of the network stack
  - Standardize ways to negotiate the obfuscation technology to use
  - Inform allied standardization work

The Pluggable Transports Community

- INTERNEWS https://internews.org
- Tor Project https://www.torproject.org/
- Guardian Project https://guardianproject.info/
- Psiphon https://psiphon.ca/
- Lantern https://getlantern.org/
- Operator Foundation https://operatorfoundation.org/
- RedJack https://www.redjack.com/
- Commercial providers (e.g., TunnelBear, ExpressVPN)

More Information on Pluggable Transports

- Best single source: https://www.pluggabletransports.info/
- PT Specification v2.0: https://www.pluggabletransports.info/spec/pt2draft3