Tranalyzer2

torDetector

Detect Tor flows

Tranalyzer Development Team

Contents

1 torDetector

1.1 Description

This plugin detects Tor flows.

1.2 Dependencies

This plugin requires the **libssl**.

Ubuntu:	`sudo apt-get install`	`libssl-dev`
Arch:	`sudo pacman -S`	`openssl`
openSUSE:	`sudo zypper install`	`libopenssl-devel`
Red Hat/Fedora[1]**:**	`sudo dnf install`	`openssl-devel`
macOS[2]**:**	`brew install`	`openssl@1.1`

1.3 Configuration Flags

Name	**Default**	**Description**
`TOR_DETECT_OBFUSCATION`	1	Detect obfuscation protocols
`TOR_DEBUG_MESSAGES`	0	Activate debug output
`TOR_PKTL`	1	Activate packet length modulo 8 heuristic

The obfuscation detection method has a pretty high rate of false positives when only one direction of the traffic is captured. It should therefore be disabled (or low confidence should be given to the resulting output) when analyzing Tor traffic which was only captured in one direction.

1.4 Flow File Output

The torDetector plugin outputs the following columns:

Column	**Type**	**Description**
`torStat`	H8	Tor status

1.4.1 torStat

The `torStat` column is to be interpreted as follows:

[1]If the `dnf` command could not be found, try with `yum` instead
[2]Brew is a packet manager for macOS that can be found here: `https://brew.sh`

torStat	Description
0x01	Tor flow
0x02	Obfuscated Tor flow (TOR_DETECT_OBFUSCATION=1)
0x04	Tor address detected
0x08	Tor pktlen modulo 8 detected
0x10	Internal state: SYN detected
0x20	Internal state: obfuscation checked
0x40	—
0x80	Packet snapped or decoding failed

1.5 Packet File Output

In packet mode (`-s` option), the torDetector plugin outputs the following columns:

Column	Type	Description	Flags
torStat	H8	Status	

1.6 Plugin Report Output

The following information is reported:

- Aggregated torStat
- Number of Tor packets

1.7 Plugin Detection Method

This subsection briefly present the detection methods used by this plugin. Tor version 0.2.7.6 and 0.4.5.10 were used to test these detection methods, they might become invalid in the future.

1.7.1 TLS Certificate

On older versions of the Tor client (< 0.2.9.15) the TLS certificate can be extracted from the traffic. The following conditions are used to determine if it belongs to a Tor flow:

Size of the certificate Tor certificates are very minimalist and are always smaller than 500 bytes.

Public key algorithm Currently Tor uses RSA-1024 certificates. Proposal 220, https://gitweb.torproject.org/torspec.git/plain/proposals/220-ecc-id-keys.txt, defines how to support Ed25519 in addition to RSA-1024. This proposal was implemented in Tor 0.2.7.5 according to the changelog: https://gitweb.torproject.org/tor.git/tree/ReleaseNotes?h=release-0.2.8#n96. However tests with Tor 0.2.7.6 (client and entry node) showed that RSA-1024 was still used for the TLS link key.

Validity period

before 0.2.4.11 Tor certificate are always valid for exactly one year (60*60*24*365 seconds).

0.2.4.11 and after In March 2013, Tor changed the way it generates the validity period in certificates. Certificates are valid for a random period of time but validity periods always start at exactly midnight.

Certificate issuer The certificate issuer is only defined by its common name (no organization, country, ...). This common name has the following format: `www.RAND.com` where `RAND` is between 8 and 20 (inclusive) base32 characters. If Tor is compiled with the `DISABLE_V3_LINKPROTO_SERVERSIDE` flag, the common name ends in `.net` instead of `.com`

Certificate subject The certificate subject is only defined by its common name (no organization, country, ...). This common name has the following format: `www.RAND.net` where `RAND` is between 8 and 20 (inclusive) base32 characters.

1.7.2 TLS Client Hello

Recent version of the Tor client (>= 0.2.9.15) use TLS 1.3 which encrypts the certificate. On these versions, only the TLS handshake Client Hello and Server Hello can be used.

Cipher list The `TLS_EMPTY_RENEGOTIATION_INFO_SCSV` is always the last element in the supported cipher list sent by Tor. Recent versions of Firefox and Chrome do not send this cipher.

TLS extensions The `renegotiation_info`, the `Application-Layer Protocol Negotiation (ALPN)` and the `Next Protocol Negotiation (NPN)` extensions are never present in Tor Client Hello messages. These extensions are almost always present in modern browsers.

Server name extension The `server_name` extension contains a hostname with the following format: `www.RAND.com` where `RAND` is between 4 and 25 (inclusive) base32 characters.

1.7.3 TLS Server Hello

The following fields are checked in the Server Hello to differentiate Tor from other TLS traffic:

TLS extensions The `Application-Layer Protocol Negotiation (ALPN)` and the `Next Protocol Negotiation (NPN)` extensions are never present in Tor Server Hello messages. These extensions are sometimes present in traffic sent by web servers.

1.7.4 Obfuscation detection

Different pluggable transport protocols can be used to obfuscate the Tor traffic between a client and a bridge. For more info: `https://www.torproject.org/docs/pluggable-transports.html.en`.

This plugin tries to detect the **obfs3** and **obfs4** obfuscation methods. The goal of these obfuscation methods is to make the traffic look completely random from the first byte (including the (EC) Diffie-Hellman key exchange). This characteristic is used to detect flows with high entropy in their first packets. Indeed, most encrypted protocols (TLS or SSH for instance) start with an unencrypted handshake phase.

1.8 Other Detection Methods

This subsection describes other possible detection methods not implemented in this plugin.

1.8.1 Relay blacklist

The list of all relays can be queried from the Tor directory. This list contain the relay IP address and OR port. Tor flows can easily be identified by comparing their IP addresses and ports against this list. A CSV list can be downloaded from `https://torstatus.blutmagie.de/`.

The disadvantages of this method are:

- It will not detect Bridge relays (`https://www.torproject.org/docs/bridges`) because it is not possible to query a list of Bridges from the Tor directory. Bridges addresses can only be retrieved three at a time from `https://bridges.torproject.org/` unless the bridge was specifically configured to not publish its descriptor.
- As this list is constantly changing, we need to regularly fetch it and keep all possible versions to use the version with the date closest to the analyzed PCAP.

The advantage of this method is:

- The traffic payload is not necessary, this method only needs the flow ports and IP addresses.

1.8.2 Packet size distribution

Tor always pack data in cells of 512 bytes. This means that when a client or a relay only has a few bytes to transmit, a packet with a size a bit superior to 512 bytes (because of TLS overhead) will be sent. This results in a peak in the packet size distribution around 550 bytes (543 bytes in the tests done with Tor version 0.2.7.6) and very few packets with smaller sizes. Figures 1 and 2 show the difference in the packet size distribution between HTTPS traffic and Tor traffic.

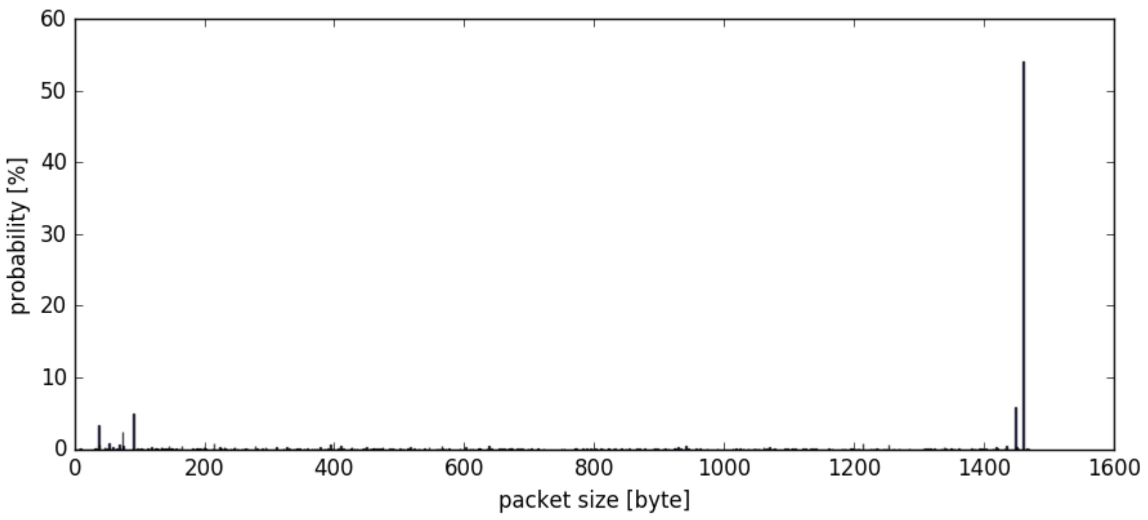

Figure 1: *Packet size distribution of HTTPS traffic.*

The disadvantage of this method is:

- False positives and negatives are way more frequent than with certificate analysis.

The advantage of this method is:

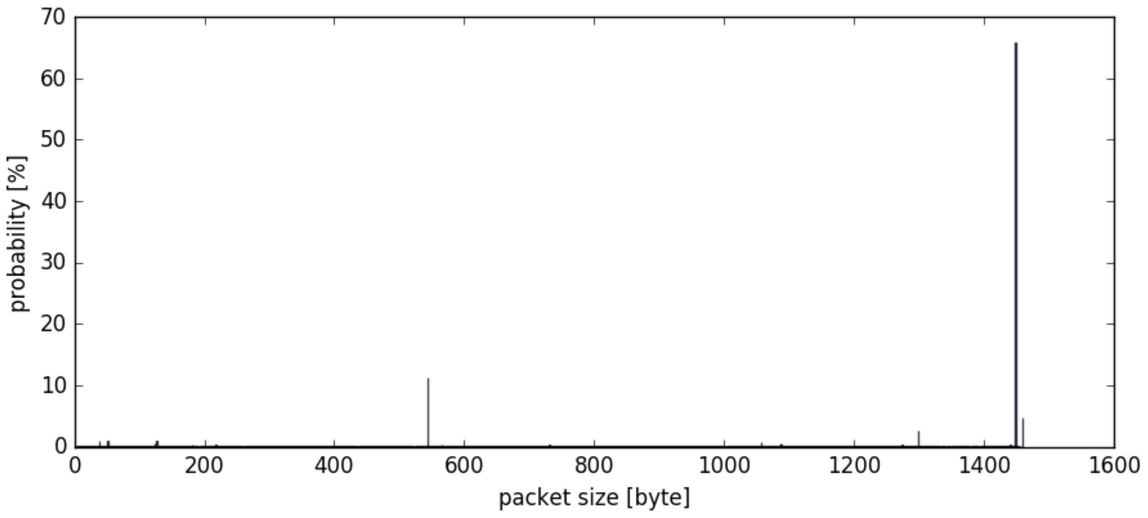

Figure 2: *Packet size distribution of Tor traffic.*

- The traffic payload is not necessary, this method only needs the packet size and the 5 (or 6 with VLAN) tuple necessary to aggregate packets in flows.