Towards a Secure and Practical System to Obfuscate Tor Network Traffic

Minjae Seo[†*], Myoungsung You[†*], Taejune Park[‡], Seungwon Shin[†], and Jinwoo Kim[§**]
[†] *KAIST,* [‡] *Chonnam National University,* [§] *Kwangwoon University*

[*] *Co-first authors,* [**] *Corresponding author*

Background and Motivation

- Tor Circuit
  - **Entry** node serves as the initial ***ingress*** point in the Tor network.
  - **Middle** node is a linking chain in the Tor network.
  - **Exit** node is the last ***egress*** point where Tor traffic hits the public Internet. It communicates directly with the server, making it capable of identifying the server's true identity and decrypting the encrypted layers of Tor traffic.
  - Given its significance, several reputable organizations already operate **trusted exit nodes** with enhanced security features and capabilities.

- ***Flow Correlation Attack (FCA)***
  - Measures the flow similarity of traversed flows observed from ***ingress*** and ***egress*** points (Figure 1)

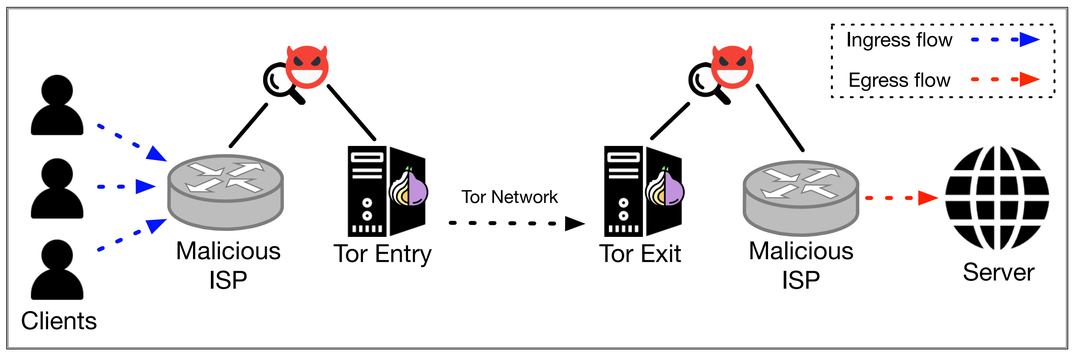

Figure 1: The main setting of a flow correlation attack

- FCA with statistical metrics
  - RAPTOR [USENIX Security'15] used the Spearman correlation coefficient, which evaluates the statistical dependence between the rankings of two given variables.

- FCA with deep learning
  - DeepCorr [CCS'18] leveraged an advanced deep learning model to learn hidden patterns of flow-level features in Tor traffic.

Limitations of Existing Solutions

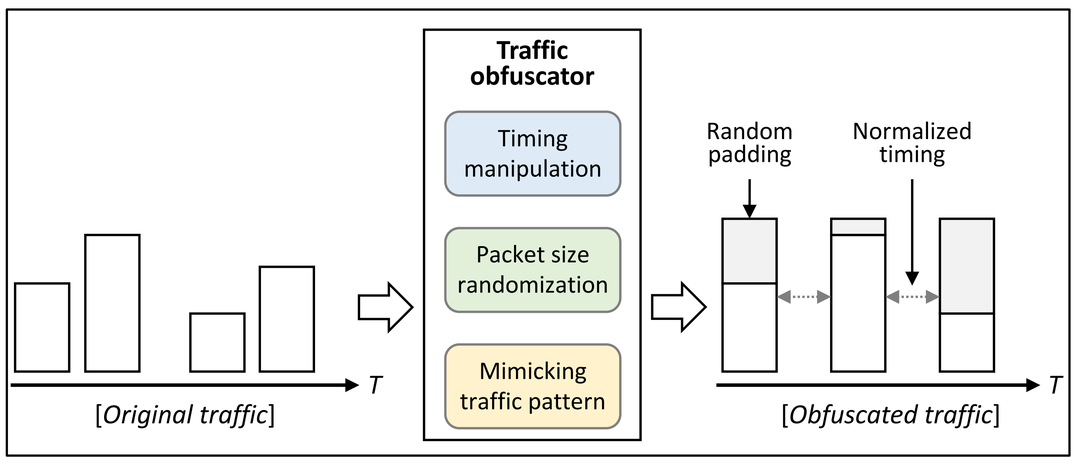

Figure 2: Common software-based traffic obfuscation methods

- CPU resource exhaustion
  - Secret Tor relay nodes (e.g., bridge nodes) or pluggable transports (e.g., meek) randomize packet bytes or disguise a Tor connection as other legitimate connections (e.g., Skype).
  - However, they rely on software-based systems to process individual packets, which **significantly consumes CPU resources**.

- Tor bandwidth waste
  - Existing countermeasures (e.g., BuFLO [S&P'12], ditto [NDSS'22]) add crafted padding bytes to the original packets and control packet transmission timing.
  - However, the inclusion of padding bytes **imposes burdens on individual packets and can waste Tor network bandwidth (see gray boxes in Figure 2)**.

System Design

- ODIN: A hardware-based Tor traffic obfuscation system (Figure 3)

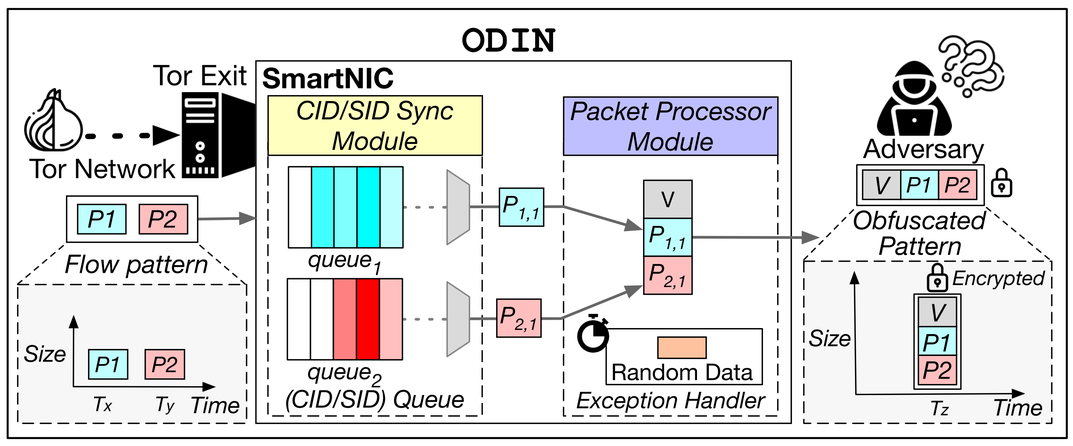

Figure 3: The system overview of **ODIN**

- ***Context-aware obfuscation***
  - The *CID/SID Sync Module* perpetually synchronizes with the circuit identifier (CID) and stream identifier (SID) information derived from the Tor application through its default API.

- ***Bandwidth-saving obfuscation***
  - The *Packet processor module* minimizes Tor bandwidth by piggybacking on packets already destined for the same final server.
  - It removes the headers of two paired packets and combines the remaining contents of both packets with a virtual header.

Evaluation

- The evaluation of ODIN primarily centers on two key aspects:
  - Its ***efficacy*** in obfuscating Tor traffic.
  - Its ***ability to impose lower overhead*** compared to an existing solution.

- Our private Tor testbed comprises three Tor container nodes (i.e., entry, middle, and exit) running on two physical machines with several services (e.g., video streaming and file downloading).

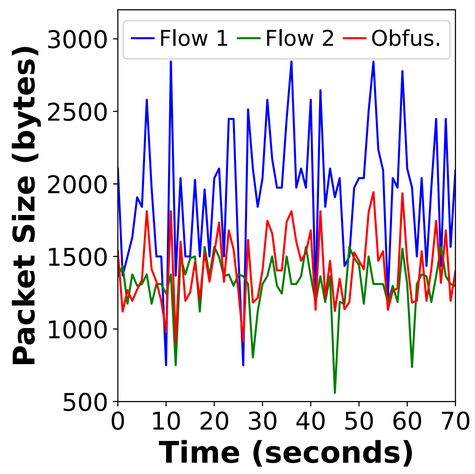

Figure 4: Traffic pattern before/after deploying ODIN

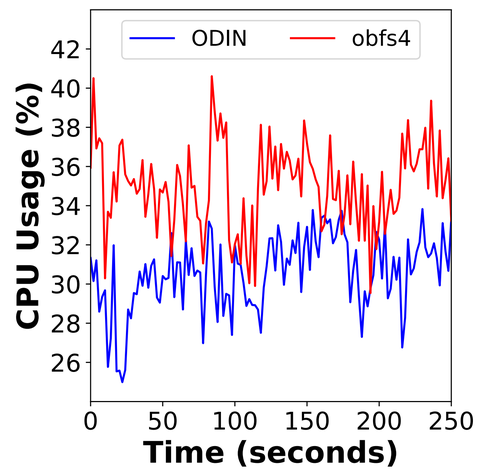

Figure 5: CPU usage of ODIN and obfs4

- Figure 4 exhibits distinct flow patterns ***without*** (i.e., Flow 1 and Flow 2) and ***with*** ODIN's obfuscation (i.e., Obfus.).

- Figure 5 presents the measured CPU usage within a specific time window, comparing the deployment of ODIN and ***obfs4***. On average, ODIN exhibits a 5% reduction in CPU overhead compared to obfs4.

Conclusion and Future Work

- We will assess the resilience of ODIN obfuscated traffic against advanced FCAs, such as deep learning-based techniques.

Acknowledgement: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government. (MSIT) (No. RS-2022-00166401, 2022R1C1C1006967).