International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Identify Encrypted Traffic of Dapps Using GNNs Using TIG

K. Prasad [1], Dr. T. Raghu Trivedi[2]

[1] M.C.A Student, Department of Computer Applications, Sri Padmavathi college of Computer science and Technology.
[2] Professor, Department of Computer Applications, Sri Padmavathi college of Computer science and Technology.

Abstract:

Decentralized programs (DApps) are increasingly advanced and deployed on blockchain structures together with Ethereum. DApp finger printing can perceive customers' visits to precise DApps via studying the ensuing network site visitors, revealing a good deal touchy information approximately the customers, such as their actual identities, economic situations and non secular or political possibilities. DApps deployed on the equal platform typically undertake the same conversation interface and comparable traffic encryption settings, making the resulting site visitors less discriminative. present encrypted traffic class techniques both require hand- crafted and nice-tuning features or suffer from low accuracy. It remains a tough assignment to conduct DApp finger printing in an accurate and efficient way. On this paper, we present Graph DApp, a unique DApp fingerprinting approach the use of Graph Neural Networks (GNNs). We propose a graph shape named visitors interplay Graph(TIG) as an data-rich representation modern-day encrypted DApp flows, which implicitly reserves a couple of dimensional features in bidirectional patron server interactions. The usage of TIG, we flip DApp finger printing into a graph classification problem and layout a effective GNN-primarily based classifier.

INTRODUCTION

The recent development of blockchain technology, the number of decentralized applications (DApps) are increasing dramatically. Unlike regular mobile or web applica- tions deployed on centralized servers, DApps run their backend codes on a decentralized peer-to-peer (P2P) network without the control of a single entity. Among the blockchain platforms on which DApps are hosted, such as EOS, NEO, Stellar, and Tron, we focus on Ethereum [1] in this paper, as it attracts the largest developer community, where more than 3,200 DApps are deployed on Ethereum and the number of daily active users has reached nearly 110,000 by 2020 [2].

The prosperity of DApps is attributed greatly to their resistance to censorship [3], [6]. Compared with the traditional mobile or web applications, DApps are completely open- sourced and autonomously managed without a single authority to manage all codes and data. Once a piece of information in DApp is added to the underlying blockchain, it gets stored permanently and thus cannot be removed or modified. In addition, blockchain technology naturally provides anonymity to each participant and thus can potentially protect identity privacy of DApp users. These features make DApps trustable to censorship and governance.

Although these service-level enchantments provide users with safety and security, the network traffic generated when individual users visit DApps can still reveal plenty of sensitive information of users. Passive adversaries (e.g., campus net- work administrators, residential network service providers, or malicious eavesdroppers) could conduct *DApp fingerprinting* to identify the specific DApps that a user visits by analyzing the resulting network traffic. An adversary can infer users' financial conditions from their usage of gambling DApps or learn their religious preferences and political views from their visits to social DApps. DApp fingerprinting can also be conducted by governors to deanonymize DApp users or even block access attempts to certain DApps without affecting visits to the rest DApps on the same platform (e.g., Ethereum) The additional contributions beyond the original paper are as follows

1) We propose Traffic Interaction Graph (TIG) to represent each individual encrypted flow, where vertices in a TIG represent packets and edges represent the packet-level interactions between a pair of client and server. We also provide quantitative measures to demonstrate the advan- tages of representing flows using TIGs over the tradi- tional packet length sequence.

2) We design GraphDApp, a powerful GNN-based classi- fier using Multi-Layer Perceptions (MLPs) and a fully- connected layer. It maps TIGs of different DApp flows to different representations in the embedding space and does not require hand-crafted features so that the clas- sification can be conducted in an effective and accurate way.

3) We collect real-world traffic datasets from 1,300 DApps on Ethereum with more than 169,000 flows. We demon- strate the accuracy and efficiency of GraphDApp in closed- and open-world settings. Compared with the state-of-the-art methods, GraphDApp has the highest classification accuracy with the shortest training time. In addition, it is also applicable to traditional mobile application

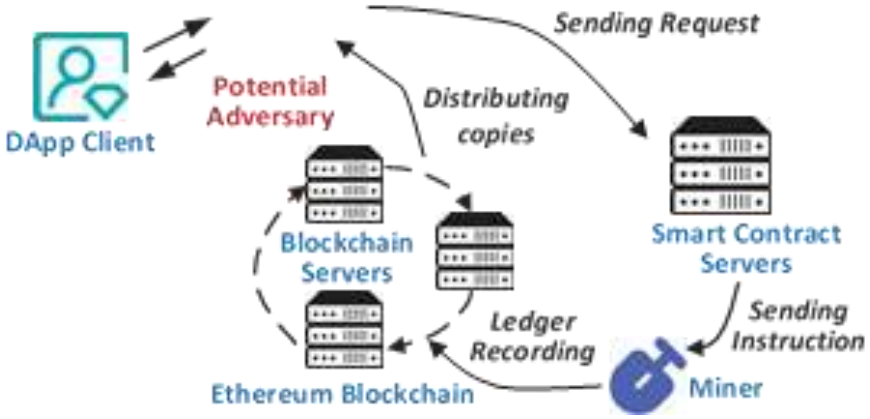


Fig. 1. Typical Workflow of DApps on Ethereum.

PROBLEM DESCRIPTION AND RELATED WORK

In this section, we describe the DApp fingerprinting prob- lem, briefly review existing studies, and emphasize the differ- ences between our work and the existing methods.

A. Problem Description

DApp fingerprinting is a traffic analysis attack that aims at identifying users' visits to a certain collection of *monitored* DApps. To be able to perform this attack, an adversary needs to observe the traffic when a victim is visiting DApps. Following the common assumption on threat model in existing literature, we assume that the potential adversary is *local* and *passive*, which means that the adversary exists in the local network of victims (e.g., campus network administrators) and performs merely traffic collection without any active attacks such as traffic hijacking, as shown in Fig. 1.

In this paper, we focus on DApps deployed on the popular platform Ethereum, as it has the largest developer community. Most DApps on Ethereum provide browser plug-ins so that they can be visited via browsers (e.g., Chrome). The frontend of a DApp implements user interfaces uniformly defined by Ethereum to render pages. The backend is *smart contracts* connecting to Ethereum blockchain network, which is com- pletely different from that of a web application that connects to a centralized database.

To initialize a visit, a DApp client (short for *client*) sends a request to the server equipped with the corresponding smart contracts, whose IP address can be obtained by DNS query. The smart contracts are the functionalities that determine the outcomes for each operation in DApp conducted by clients. All data records of these operations and outcomes are then packaged in the form of *blocks* by the miners on the underlying blockchain platform and stored on the distributed ledger. The client also gets a list of blockchain servers from the smart contract server, through which it can acquire updated information from the ledger to render webpages.

SSL/TLS protocols are used to encrypt the transmission data between clients and the blockchain platform. There are mainly two layers: the Handshake layer and the Record layer . The former is used to negotiate secure parameters of an SSL/TLS session, while the latter is responsible for transferring encrypted data under the secure parameters.

Different from traditional mobile and web applications, DApps on Ethereum implement the same frontend interface, adopt similar settings in SSL/TLS implementation, running their backend codes and storing their data in the same decen- tralized blockchain network. These common features make the resulting traffic of different DApps less discriminative

Algorithm 1 Construction of Traffic Interaction Graph

Input: A packet length sequence $P\ (p_1, \ldots, p_N)$

Output: The corresponding traffic interaction graph $G\ (V, E)$

1: Initialize V and E as empty sets

2: **for** $p_i \in P$ **do**

3:Add a vertex with a length value of p_i in V

4: Separate V into bursts $B\ (b_1, \ldots, b_K)$ according to packet direction

5: **for** $b_i \in B$ **do**

6: **if** *len*(b_i)> 1 **then**

7: **for** $v_j \in b_i$ **do**

8: Add an edge between v_j and v_{j+1} in E

9: **for** $b_i \in B$ **do**

10: **if** *len*(b_i) 1 and *len*(b_{i+1}) 1 **then**

11: Add an edge between b_i and $b_{i\ 1}$ in E

12: **else**

13: Add two edges between b_i and b_{i+1} in E

14: **return** G

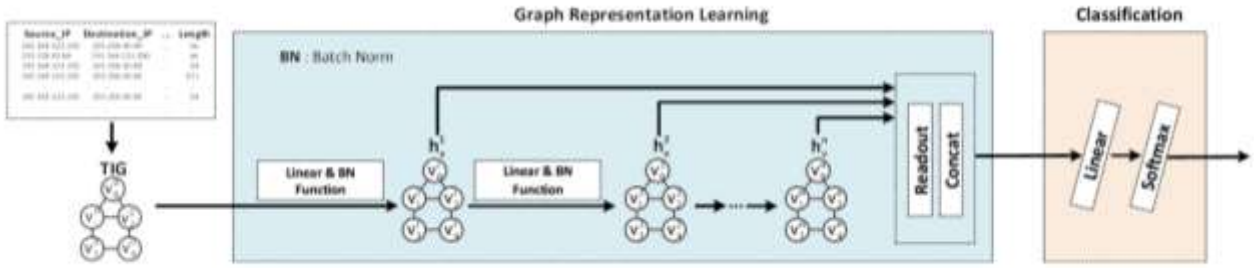
A. TIG Samples of DApps

With an intuitive description of TIG in the last subsec- tion, we formally present the construction process of TIG, as depicted in Algorithm 1.

The construction process considers the packet length sequence of a specific flow as the input and starts with an initialization of the vertex set V and edge set E . Then, it constructs the vertex set (lines 2-3) and obtain the bursts (line 4). After that, it iteratively adds intra-burst edges (lines 5-8) and inter-burst edges (lines 9-13). Finally, the resulting G is the desired TIG.

To get a better understanding of the representation capacity of TIG, we select 3 kinds of DApps (i.e., Aigang, Aragon and AaveProtocol) and visualize the corresponding TIGs constructed with Algorithm 1, as shown in Fig. 3. Since it is impossible to enumerate the TIGs for all the flows, we randomly select a flow for each DApp and use Matplotlib

[4] to draw the corresponding TIG. It is clear to observe the differences in terms of graph structure: Aigang owns spindle- shaped TIGs for its flows, Aragon has simple and compact TIGs, while AaveProtocol exhibits complex and fish-shaped TIGs.



Structure of Neural Networks in Graph DApp.

B. The Benefits of TIG

The following explains why TIG is a powerful representa- tion of DApp flows. In general, TIG can extract features of DApp flows from four aspects, each of which has been proven to be valuable for traffic classification.

1) **Packet direction information.** The direction informa- tion is revealed by the sign of the vertex in TIG, where positive values indicate downstream packets while negative for upstream packets.

2) **Packet length information.** The packet length sequence, as well as its mathematical variants, are oftenused as the key features in encrypted traffic classification [17]. Vertices in TIG are associated with the corre- sponding packet lengths, which can be naturally used by classifiers.

3) **Packet burst information.** The vertices of the same layer in TIG represent the packets composing an indi- vidual burst. The burst-level behaviors for different applications can vary significantly and thereby act as discriminative features learned by classifiers.

4) **Packet ordering information.** TIG can represent the order of packets from the beginning of SSL/TLS session negotiation to the end of application data transmission. In addition, TIG also reflects the interaction between the server and the client.

CONCLUSION

In this paper, we proposed GraphDApp, which can identify encrypted traffic of DApps using GNNs. We constructed the TIGs of encrypted flows based on the packet length and packet direction and turned the DApp identification into a graph clas- sification problem. Then, we employed multi-layer

perceptions to build a GNN-based classifier. Experiments are conducted to evaluate our method on real traffic datasets collected from DApps on Ethereum, including 40 monitored DApps and 1,260 unmonitored DApps. The experimental results show that GraphDApp can improve the accuracy by at least 10% over the state-of-the-art. We also demonstrated the effectiveness of GraphDApp on mobile application fingerprinting. In the future

REFERENCES

[1]. Ethereum. Accessed: Oct. 1, 2019. [Online]. Available: https://www. ethereum.org/

[2]. State of the Dapps. Accessed: Oct. 1, 2019. [Online]. Available: https://www.stateofthedapps.com/dapps?page=1

[3]. Countries and Territories. Accessed: Oct. 5, 2019. [Online]. Available: https://freedomhouse.org/countries/freedom-world/scores

[4]. Matplotlib. Accessed: Oct. 20, 2019. [Online]. Available: https:// matplotlib.org/

[5]. Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, and P. Martineau, "An exact graph edit distance algorithm for solving pattern recognition problems," in Proc. Int. Conf. Pattern Recognit. Appl. Methods, Lisbon, Portugal, vol. 1, Jan.2015, pp. 271–278.

[1]. S. Feghhi and D. J. Leith, "A Web traffic analysis attack using only timing information," IEEE Trans. Inf. Forensics Security, vol. 11, no. 8, pp. 1747–1759, Aug. 2016.

[2]. E. Grolman et al., "Transfer learning for user action identication in mobile apps via encrypted trafc analysis," IEEE Intell. Syst., vol. 33, no. 2, pp. 40–53, Mar. 2018.

[3]. J. Hayes and G. Danezis, "K-fingerprinting: A robust scalable website fingerprinting technique," in Proc. 25th USENIX Secur. Symp., Austin, TX, USA, vol. 16, Aug. 2016, pp. 1187–1203.

[4]. K. Hornik, "Approximation capabilities of multilayer feedforward net- works," Neural Netw., vol. 4, no. 2, pp. 251–257, 1991.

[5]. K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Netw., vol. 2, no. 5, pp. 359–366, Jan. 1989.

[6]. M. Korczynski and A. Duda, "Markov chain fingerprinting to classify encrypted traffic," in Proc. IEEE Conf. Comput. Commun., Toronto, ON, Canada, Apr. 2014, pp. 781–789.

[7]. M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for HTTPS and QUIC," in Proc. IEEE Conf. Comput. Commun., Honolulu, HI, USA, Apr. 2018, pp. 1331–1339.

[8]. A. Panchenko et al., "Website fingerprinting at Internet scale," in Proc. Netw. Distrib. Syst. Secur. Symp., San Diego, CA, USA, Feb. 2016, pp. 1–5.

[9]. S. Rezaei and X. Liu, "Deep learning for encrypted traffic classifica- tion: An overview," IEEE Commun. Mag., vol. 57, no. 5, pp. 76–81, Dec. 2019.

[10]. V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in Proc. Netw. Distrib. Syst. Secur. Symp., San Diego, CA, USA, Feb. 2018, pp. 1–15.