Poster: User Sessions on Tor Onion Services: Can Colluding ISPs Deanonymize Them at Scale?

Daniela Lopes
INESC-ID / Instituto Superior Técnico, Universidade de Lisboa
daniela.lopes@tecnico.ulisboa.pt

Pedro Medeiros
INESC-ID / Instituto Superior Técnico, Universidade de Lisboa
pedro.de.medeiros@tecnico.ulisboa.pt

Jin-Dong Dong
Carnegie Mellon University
jd0@cmu.edu

Diogo Barradas
University of Waterloo
diogo.barradas@uwaterloo.ca

Bernardo Portela
INESC TEC / Universidade do Porto
bernardo.portela@fc.up.pt

João Vinagre
INESC TEC / Universidade do Porto
jnsilva@inesctec.pt

Bernardo Ferreira
LASIGE, Faculdade de Ciências, Universidade de Lisboa
blferreira@fc.ul.pt

Nicolas Christin
Carnegie Mellon University
nicolasc@cmu.edu

Nuno Santos
INESC-ID / Instituto Superior Técnico, Universidade de Lisboa
nuno.m.santos@tecnico.ulisboa.pt

ABSTRACT

Tor is the most popular anonymity network in the world. It relies on advanced security and obfuscation techniques to ensure the privacy of its users and free access to the Internet. However, the investigation of traffic correlation attacks against Tor Onion Services (OSes) has been relatively overlooked in the literature. In particular, determining whether it is possible to emulate a global passive adversary capable of deanonymizing the IP addresses of both the Tor OSes and of the clients accessing them has remained, so far, an open question. In this paper, we present ongoing work toward addressing this question and reveal some preliminary results on a scalable traffic correlation attack that can potentially be used to deanonymize Tor OS sessions. Our attack is based on a distributed architecture involving a group of colluding ISPs from across the world. After collecting Tor traffic samples at multiple vantage points, ISPs can run them through a pipeline where several stages of traffic classifiers employ complementary techniques that result in the deanonymization of OS sessions with high confidence (i.e., low false positives). We have responsibly disclosed our early results with the Tor Project team and are currently working not only on improving the effectiveness of our attack but also on developing countermeasures to preserve Tor users' privacy.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; **Pseudonymity, anonymity and untraceability**; • **Networks → Network privacy and anonymity**; • **Information systems → Traffic analysis**.

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11.
https://doi.org/10.1145/3548606.3563520

KEYWORDS

Traffic Analysis; Tor; Onion Services; Flow Correlation Attacks; Anonymous Communications

ACM Reference Format:
Daniela Lopes, Pedro Medeiros, Jin-Dong Dong, Diogo Barradas, Bernardo Portela, João Vinagre, Bernardo Ferreira, Nicolas Christin, and Nuno Santos. 2022. Poster: User Sessions on Tor Onion Services: Can Colluding ISPs Deanonymize Them at Scale?. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3548606.3563520

1 INTRODUCTION

The Tor anonymity network is currently used by a large community of Internet users. Many individuals such as journalists, activists, and common citizens rely on Tor as a fundamental privacy-enhancing tool to anonymously access the web and circumvent eavesdropping, surveillance, and censorship. Moreover, Tor can also be used for deploying Onion Services (OSes) to provide an infrastructure for delivering anonymous, censorship-resistant online services such as whistleblowing websites and news outlets. By hiding the IP addresses of the user accessing the service and of the OS itself, OSes preserve both sender and receiver anonymity. Over the years, the popularity of Tor has fueled extensive research aimed at securing the user community against various attacks. However, important questions remain understudied about the potential deanonymization of Tor OS traffic.

First, *is it possible to fully deanonymize OS user sessions through traffic correlation attacks?* OS website fingerprinting attacks [4, 6] allow the tracking of clients accessing a given OS, but they cannot deanonymize the IP address of the OS operator. Prior work on traffic correlation attacks based on deep learning classifiers [5, 7] focused primarily on deanonymizing regular Tor circuits (e.g., clients accessing the open web through Tor). However, by conducting extensive experimentation in a controlled environment, we found that these techniques alone are difficult to scale due to the high overhead

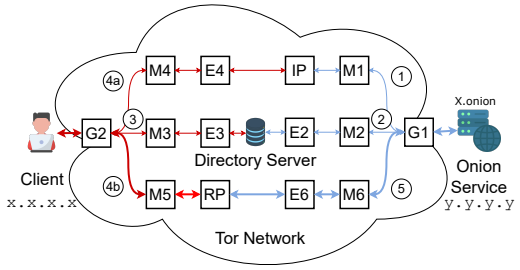

Figure 1: OS session between a client and an OS: When starting up, the OS recruits random Tor nodes as *introduction points* (IP) (1) and publishes its onion address and IP identifier in the directory server (2). The client can then perform an address lookup to learn the IP identifier (3), choose any Tor relay as the *rendezvous point* (RP), and establish a circuit with it (4b). It also establishes a circuit to the OS IP to let the OS know it wants to connect to the RP (4a). The OS can now establish a circuit to the RP chosen by the client (5).

introduced by convolutional neural networks. Moreover, their accuracy is dramatically degraded in virtue of the higher complexity of the circuits established between clients and OSes. As shown in Figure 1, the circuits established between clients and OSes introduce extra hops and higher latency, translating into different traffic patterns, lowering the precision of the machine learning models. Nonetheless, and even though limited, these techniques show that full deanonymization attacks against OSes may be possible.

Second, *can OS traffic deanonymization attacks be launched in the real world at scale?* If said attacks do exist and can plausibly be launched globally, then the Tor OS users and providers face a real danger of having their privacy compromised. An essential question can then be posited on how many real-world ISPs would need to collude and how geographically spread would they need to be to intercept a sufficiently large fraction of the Tor OS traffic and approximate the capabilities of a global passive adversary.

Third, *are there any viable countermeasures to thwart these attacks and to protect the Tor OS user community?* Ideally, such defensive techniques should be easy to deploy, preferably without requiring any changes to Tor's existing core protocols, software, or infrastructure.

In this paper, we present ongoing work toward addressing these research questions. In particular, we give an overview of our technical approach to overcome the scalability and accuracy limitations of existing traffic correlation techniques and provide a few preliminary results that suggest that our attacking techniques can be effective (albeit ensuring that no sufficient details are given before fully devising proper countermeasures). We have responsibly disclosed the early results of our work with the Tor development team, and we will continue to do so in the future as we keep improving our attacking techniques and develop countermeasures.

2 TECHNICAL APPROACH

To investigate the feasibility of deanonymizing Tor OS sessions, our approach is to devise a distributed attack involving a group of colluding ISPs whose role is to intercept and monitor the network traffic of Tor guard nodes (e.g., nodes G1 and G2 in Figure 1). By performing a longitudinal measurement of the Tor guard probability statistics taking a snapshot of Tor relays' details using the Onionoo service [8], we found that a relatively small number of instrumental ISPs would suffice to monitor a large fraction of Tor traffic. From our analysis, as few as six different ISPs are enough to observe more than 50% of all traffic in the Tor network. This effect is due to the skewed distribution of Tor guards towards certain geographic regions, most notably within Europe.

The idea is then to rely on network monitoring nodes (aka *filtering nodes*) deployed by colluding ISPs, which obtain and share meta-data of Tor traffic samples collected at various vantage points. ISPs can then submit deanonymization queries to a set of computation servers (*matching nodes*) collectively managed by the ISPs themselves or by a third party (e.g., a cloud provider). Matching nodes analyze the traffic collected by the ISPs and identify samples that belong to the same OS sessions, reporting the deanonymized IP addresses of the session endpoints, i.e., the IP addresses `x.x.x.x` and `y.y.y.y` in the example of Figure 1.

Together, filtering nodes and matching nodes implement our traffic correlation technique which consists of a 5-stage pipeline as shown in Figure 2. The filtering phase will be applied by filtering nodes to ignore irrelevant traffic samples and to organize them into the features required by the matching nodes. It is composed of: i) the origin checker receives the raw traffic captures and separates them into OS traces and client traces; ii) the request separator splits the captures into the multiple requests that compose them, and iii) the OS request identifier filters client requests to OSes. The matching phase, carried out by the matching nodes, correlates the requests issued by a client to an OS and groups them into the whole session performed. It is composed of: i) the request correlator, which finds the OS request associated with a given client request; and ii) the session assembler that groups correlated requests into sessions.

The main novelty in our pipeline lies in the request correlator stage (i.e., stage four). In contrast to the state-of-the-art [5, 7], we are exploring alternative approaches that are both more efficient and accurate than deep learning classifiers for correlating requests within Tor OS sessions. In one of our most promising attempts so far, we start by fixing the initial and final absolute times of a client request and split the received packets during this time interval into buckets of 500 ms each. We do the same splitting for the packets sent on the OS side and apply an adapted version of the known NP decision problem Subset Sum [10]. Since we apply it on a bounded time series, it is particularly effective at matching patterns between the packets received by the clients and packets sent by the OS. For now, our algorithm checks if there is a combination of packets sent by the OS that approximately matches the total packets received by the client. We then improve the request correlator precision by grouping multiple requests into corresponding sessions in the session assembler. For instance, we may require at least 3 requests from the same session to be considered correlated by our request correlator to assume we indeed found a correlated session between a client and an OS.

3 PRELIMINARY RESULTS

We have built an initial prototype using Python. To implement the origin checker and the OS request identifier filtering stages, we draw inspiration from earlier encrypted traffic analysis tasks in

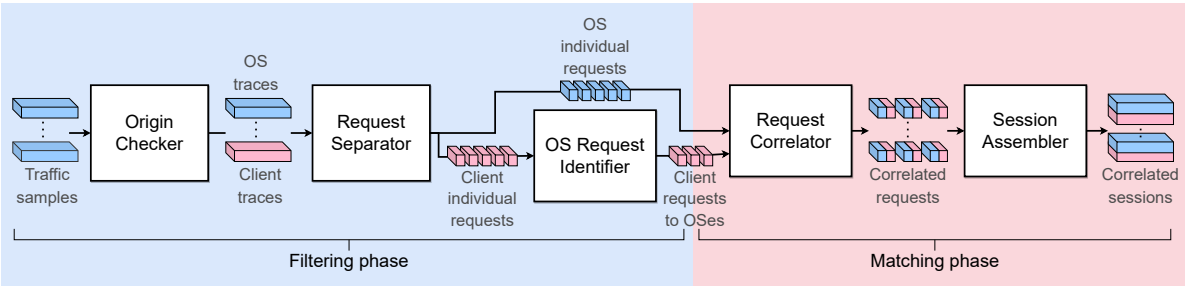

Figure 2: Traffic deanonymization pipeline.

related domains [1, 3] and make use of XGBoost [9]. To evaluate our prototype, we generated a dataset containing the traces of traffic that we emulated between several clients and OSes under our control in a laboratory environment. We set up 7 virtual machines (VMs) hosting OSes and 20 VMs acting as Tor clients, scattered in different locations across the globe. To generate a set of OS interactions that follows realistic popularity rates, we considered the access rates to the top 7 most popular OSes [2] – excluding those identified as botnet C&C servers. To make our selection of representative webpages, we created 7 webpages by crawling *ahmia.fi*, a clearnet search engine for Tor OSes. We emulated a set of concurrent browsing sessions on our OSes, in which each client placed 60 sessions, each session targeted to an OS chosen probabilistically according to its popularity. Each session includes 5 requests to the webpage hosted by the OS.

Figure 3 shows the variation of precision and recall when grouping the requests into sessions of at least 1, 2, 3, 4, or 5 requests for a dataset with 268031 possible sessions, where only 1020 of those are correlated. Our preliminary results of the matching phase using ground truth are encouraging, showing that our solution is 3 orders of magnitude faster than existing traffic correlation techniques on Tor traffic [5] and it achieves 100% precision and 97.7% recall when leveraging at least 4 requests of the same session. By requiring that a higher number of requests to be correlated within a given session, e.g., five requests rather than four, the number of false negatives increases causing a reduction in the recall. On the other hand, the number of false positives decreases making the results of the obtained correlated sessions more reliable.

Our current implementation still has some drawbacks, such as not accounting for situations with timing incoherences, where the client may receive most of the packets before the OS sent them, or much after, which do not occur in a significant number of cases for our synthetic datasets but is bound to occur more often in the wild. So, our next steps will consist of establishing a timing relationship between the packets observed in each bucket at the client and the OS side, for instance with frequency domain signal processing and analysis. We will thoroughly evaluate our techniques using richer datasets and develop countermeasures based on our findings. We will also make a direct comparison of the performance and scalability of our solution against the state-of-the-art DeepCoFFEA [7] at deanonymizing OS traces, and provide further insight on why our Subset Sum parameterization works so well for this kind of trace.

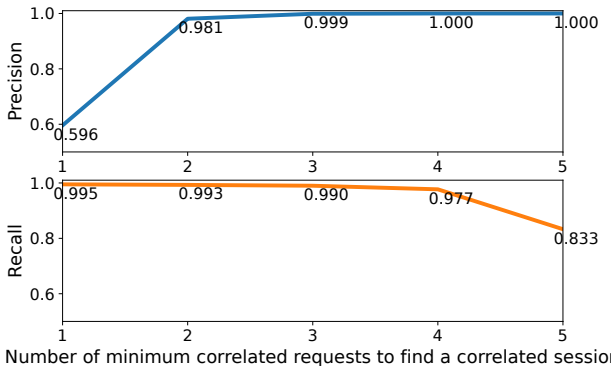

Figure 3: Precision and recall variation when grouping multiple requests into a session.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments and insightful feedback. This work was supported by the Fundação para a Ciência e Tecnologia (FCT) under grants UIDB/50021/2020 and (DAnon) CMU/TIC/0044/2021.

REFERENCES

[1] Diogo Barradas, Nuno Santos and Luís Rodrigues. 2018. Effective Detection of Multimedia Protocol Tunneling using Machine Learning. In *Proc. of USENIX Security'18.*
[2] Gareth Owenson and Nicholas Savage. 2016. Empirical analysis of Tor hidden services. *IET Information Security* (2016).
[3] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *Proc. of USENIX Security'16.*
[4] Jiajun Gong and Tao Wang. 2020. Zero-delay Lightweight Defenses against Website Fingerprinting. In *Proc. of USENIX Security'20.*
[5] Milad Nasr, Alireza Bahramali and Amir Houmansadr. 2018. DeepCorr: Strong Flow Correlation Attacks on Tor Using Deep Learning. In *Proc. of CCS'18.*
[6] Rebekah Overdorf, Marc Juarez, Gunes Acar, Rachel Greenstadt and Claudia Diaz. 2017. How Unique is Your .Onion? An Analysis of the Fingerprintability of Tor Onion Services. In *Proc. of CCS'17.*
[7] Se Oh, Taiji Yang, Nate Mathews, James Holland, Mohammad Rahman, Nicholas Hopper and Matthew Wright. 2022. DeepCoFFEA: Improved Flow Correlation Attacks on Tor via Metric Learning and Amplification. In *Proc. of S&P'22.*
[8] The Tor Project. 2021. Onionoo. https://metrics.torproject.org/onionoo.html. Accessed: 2022-08-14.
[9] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proc. of KDD'16.*
[10] Vitor Curtis and Carlos Sanches. 2016. An Efficient Solution to the Subset-Sum Problem on GPU. *Concurrency and Computation: Practice and Experience* (2016).