Exposing the Rat in the Tunnel: Using Traffic Analysis for Tor-based Malware Detection

Priyanka Dodia
Qatar Computing Research Institute
Qatar
pgdodia@hbku.edu.qa

Mashael AlSabah*
Qatar Computing Research Institute
Qatar
msalsabah@hbku.edu.qa

Omar Alrawi
Georgia Institute of Technology
USA
alrawi@gatech.edu

Tao Wang
Simon Fraser University
Canada
taowang@sfu.ca

ABSTRACT

Tor [31] is the most widely used anonymous communication network with millions of daily users [6]. Since Tor provides server and client anonymity, hundreds of malware binaries found in the wild rely on it to hide their presence and hinder Command & Control (C&C) takedown operations. We believe Tor is a paramount tool enabling online freedom and privacy, and blocking it to defend against such malware is infeasible for both users and organizations.

In this work, we present effective traffic analysis approaches that can accurately identify Tor-based malware communication. We collect hundreds of Tor-based malware binaries, execute and examine more than 47,000 active encrypted malware connections and compare them with benign browsing traffic. In addition to traditional traffic analysis features (which work at the connection level), we propose global host-level network features to capture peculiar malware communication fingerprints across host logs. Our experiments confirm that our models are able to detect "zero-day" malware connections with 0.7% FPR even when malware connections constitute less than 5% of Tor traces in the test set. Using multi-labeling approaches, we are able to accurately detect the malware behavior-based classes (grayware, ransomware, etc). Finally, we evaluate the robustness of our models on real-world enterprise logs and show that the classifiers can identify infected hosts even with missing features.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; **Privacy-preserving protocols**.

KEYWORDS

Malware; Tor; Traffic analysis

*Joint first author

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

ACM ISBN 978-1-4503-9450-5/22/11…$15.00
https://doi.org/10.1145/3548606.3560604

ACM Reference Format:
Priyanka Dodia, Mashael AlSabah, Omar Alrawi, and Tao Wang. 2022. Exposing the Rat in the Tunnel: Using Traffic Analysis for Tor-based Malware Detection. 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, 16 pages. https://doi.org/10.1145/3548606.3560604

1 INTRODUCTION

Botnets and malware hiding behind Tor [31] have been reported almost a decade ago [1, 29]. More recently, off-the-shelf tools sold in underground forums, such as SystemBC [18], are increasingly popular in aiding cyber criminals to build and launch Tor-based malware. Malware-infected machines often need to connect to external servers to communicate with their C&C, fetch ransomware payment pages, or download files needed for their operations [25]. Such malware can hide such activities and evade detection by exploiting the client- and server-side anonymity guarantees provided by Tor. With the hundreds of Tor-based malware being launched in the wild on a daily basis, there is an increasing need to detect such malware. This will not only protect enterprise hosts from the growing threat, but it will also rid the Tor network of the overload of bots.

Since Tor router IPs are published publically, one solution is to block connections. However, for individuals and enterprises, there are various legitimate use cases for VPNs and Tor. Traditional malware detection techniques based on destination IP addresses, ports, or Deep Packet Inspection (DPI) are increasingly ineffective due to the use of Tor routers between infected machines and their C&C or destinations. Instead of directly connecting to enumerable C&C IPs, multiple infected machines in a network will instead appear as connecting to multiple Tor routers in network logs.

We believe that traffic analysis is a promising approach to address this problem. Traffic analysis is the process of examining traffic patterns (packet sizes, directions, timings, etc) to infer more (sensitive) information about traffic, thereby reducing the expected privacy provided by encryption or by proxy-based anonymization. For example, traffic analysis on encrypted communication can identify the language spoken in a VOIP call [68], sensitive webpages visited (healthcare, legal, etc) [45], and Netflix show streamed [58]. In the area of anonymous communication, traffic analysis has shown success in a wide range of applications, mostly related to censorship (detecting obfuscated communication) and privacy attacks [54]. For

example, Website Fingerprinting (WF) [27, 36, 43, 53, 64, 65] can identify a webpage visited by a Tor client only by using a supervised machine learning classifiers pre-trained with traffic features extracted from prior visits to the same webpage.

In the realm of malware detection, traffic analysis has different design goals that introduce new challenges. In WF, for example, traffic analysts may not be interested in identifying uncensored pages, whereas, for us, it is essential to detect zero-day malware. Second, creating large-scale ground-truth datasets of real malware connections is significantly more challenging than in other domains. Another fundamental challenge is that we have to differentiate between benign and malicious Tor connections so that we do not interrupt the use of Tor for legitimate users.

In this paper, we present the first traffic analysis approach to defend against Tor-based malware. We show that it is possible to accurately differentiate between benign and malicious encrypted Tor connections. More importantly, we show that this can be done for "zero-day" malware variants, which have not been seen by our models. This is achieved by applying traffic analysis techniques to encrypted malware traffic and logs. We observe that Tor-based malware exhibits execution or connection patterns that deviate from legitimate or benign Tor-based browser use. We believe that the analysis of encrypted traffic patterns can automate the identification of such malware.

Our contributions. First, we build and update a repository consisting of hundreds of thoroughly verified fresh malicious Tor binaries. We inspect their traffic and ensure they are indeed malicious and use Tor for their operation. For creating a benign dataset, we build various binaries that simulate different profiles of traffic browsing (using the Tor browser) with different settings and loads. We deploy our (benign and malware) binaries independently on a sandbox environment and collect thousands of traffic instances over months. We also incorporate Tor traffic instances generated using various applications that were independently collected and published. We make our datasets available to the research community (See Section 2.3).

Second, we characterize Tor-based malware binaries through the analysis of different families and network traffic exchange. We identify several characteristics that differentiate between malicious Tor communication and typical benign Tor browsing sessions. We craft and extract novel combinations of connection- and host-level traffic features from benign and malicious traffic. We then create (1) binary classifiers that can distinguish malware from benign connections, and (2) multi-label malware classifiers that can identify the malware class (ransomware, worm, trojan, etc.) based on their Tor communication.

Finally, we craft several experiments to demonstrate the usefulness of our approach, feature categories, and models. In testing our models, we build datasets that reflect realistic scenarios where malware connections comprise a minimal percentage of all Tor connections. Our experiments show that we can identify the following: (1) malware connections, (2) malware class, and (3) zero-day malware. Furthermore, we experiment on real-world enterprise network logs which we obtain from our partners, and we are able to red flag suspicious connections despite missing some features due to the unavailability of raw PCAPs.

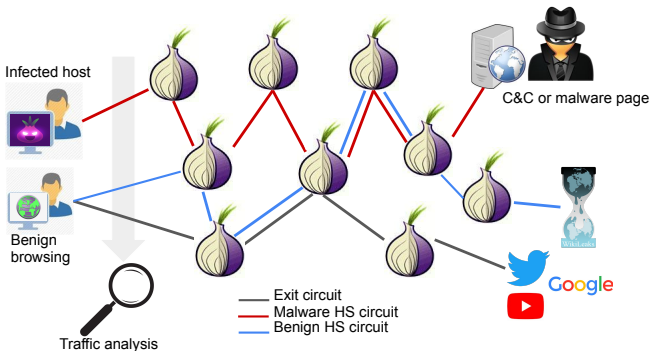

Figure 1: Structure of malware and benign circuits, and where traffic analysis is performed in Tor

Results summary. Our experiments, which are based on more than 47,000 active real Tor binary connections, show that our models are able to identify malware traffic with 93.3% precision, 81.6% recall, and 0.88% FPR. We further challenge our models by introducing unseen "zero-day" binaries and show that we are able to correctly identify *all* malware connections with 0.7% FPR even when malware instances comprise 5% of the test data. Our multi-label classification models are able to predict the malware class with high precision and low hamming loss. Finally, our real-world enterprise logs experiment indicates that our classifiers are able to identify the infected host even with missing features.

2 BACKGROUND AND MOTIVATION

2.1 Tor and traffic analysis

It is no surprise that an increasing number of malware variants leverage Tor to hide their presence. To get client anonymity, users typically install the Tor browser, which runs the *Onion Proxy (OP)* behind a modified Firefox browser. OPs tunnel users' traffic to their destinations through *circuits*, paths consisting of three *Onion Routers (ORs)* that are volunteer operated from all over the world. The three ORs in a circuit are known as the *entry guard*, *middle*, and *exit*. ORs and OPs share the network information including the OR names, IPs, connection ports, public keys, and other information through the router *consensus* document. Using the public keys in the consensus, ORs establish TCP connections secured by TLS, and each connection multiplexes multiple circuits. Tor sends data in fixed-sized onion-encrypted units known as *cells*.

Tor also provides server anonymity. This is known as *onion* or *hidden services (HS)*. In this process, a user can deploy a server and serve content without revealing her IP address or location. Onion domain names appear random and end with *.onion*. Many legitimate popular domains such as Facebook, duckduckgo, and WikiLeaks operate as HS. However, the anonymity provided by HS makes it also attractive for malware as a hideout for C&C servers, which hinders takedown operations. As shown in Figure 1, Tor-based malware connects to C&C through a 3-hop exit circuit if the external server is deployed on a publically accessible machine, or through 6-hop HS circuits if the external server is hosted as a hidden or onion service.

Table 1: Comparison to existing C&C traffic detection solutions

Related work	Detection point	Detection artifact	Detection approach	Scope	Class detection
BotMiner [35]	Network (client side)	TCP/UDP flow size DNS, SMTP, C&C IP	Unsupervised network flow clustering	Coordinated bots	✗
Jackstraws [40]	End Host	System calls	Supervised system call behaviour graph clustering	Generic malware	✗
TorWard [42]	Network (Tor exit OR)	TCP flow DPI, DNS C&C IP	Signature-based DPI	Tor exit traffic abuse	✓
BOTection [22]	Network (client side)	TCP/UDP/ICMP connection state, protocols (eg. DNS)	Connection state stochastic modeling	Bots w/ bursty connection behavior	✓
This work	Network (client side)	Tor cell sequences (TCP), connection states, DNS	Traffic analysis on encrypted flows	Tor-based malware	✓

For a user or malware to reach HS, either (1) an OP (sometimes modified or stale) is shipped with the binary and installed, or (2) Tor2Web is used. Tor2web is a service that allows users to access HS without installing the Tor OP.[1]

The threat model of Tor assumes a partial-view active adversary, who can control or view parts of the network, and generate, edit or delay traffic. If an attacker monitors both ends of a circuit (OP to entry guard and exit to destination server), he will be able to link the source to the destination using traffic or timing analysis and thereby break anonymity. A dangerous class of traffic analysis attacks known as Website Fingerprinting (WF) challenges this traditional threat model as it only requires the attacker to be present between the OP and the entry guard. Figure 1 illustrates the point where traffic analysis and WF are carried out in Tor. By analyzing the encrypted traffic, the attacker is able to identify the visited page and link the user to her destination.

Traffic analysis, and specifically WF, is carried out as follows. First, the attacker visits his target pages using Tor and collects network traces. Next, he extracts distinguishing features (such as the sequences of packets or timings between packets or bursts, etc) and uses them to train a supervised classifier. Later, when the victim visits a page, the attacker uses the trained model to classify the traces to a website. In this work, we follow the same approach where we collect traffic of (1) various malware binaries and (2) benign browsing sessions, and train a classifier to identify the traffic of malware binaries, even those never previously seen by the classifier. Section 9 summarizes prior WF work and compares it to our work.

2.2 Motivation: Enterprise logs case study

To illustrate and motivate our problem clearly, we consider a malware tracking case study from real-world enterprise logs that were shared by our partners. The data contains large volumes of enterprise network traffic captured with BRO/Zeek logs [21] (connection, DNS, HTTP, SSL, etc.) from May 2018 to February 2019. The data contains more than 600 million entries in the connection log for TCP, UDP, and ICMP connections and more than 170 Million entries in the DNS logs.

Investigating the DNS logs for suspicious activities revealed interesting observations. We identified several peculiar access attempts to onion domains. Under correct usage of the Tor browser, onion domains should not be visible in the DNS logs, as they are resolved within the Tor network for anonymity reasons. Upon close inspection, some of these domains were based on Tor2Web, and some appeared to be benign domains possibly browsed by users on a standard non-Tor browser (leading to their appearance in DNS logs). However, seven onion domains[2] stood out due to the high frequency of their request rate in the logs.

These are related to the `WannaCry` ransomware, which is known to use Tor to communicate with its C&C. Further inspection of the connection logs revealed that the same IP that requested the `WannaCry` onion domains also generated 197 Tor connections. Studying DNS logs also revealed lookups for *kill switch* domains. These are domains that are hardcoded in the malware to regulate its operation. Figure 6 (in Appendix A) shows the timestamps of Tor connection attempts, kill switch, and WannaCry onion domain accesses. We revisit this case study in Section 8 and show how our models can identify malicious connections without relying on DNS onion leaks.

Challenges in detecting Tor-based malware. In this particular incident, onion accesses coincided with the Tor connection establishment attempts and originated from the same source, which led us to believe some of the Tor connections may be related to the malware (malware tries to connect to its C&C through Tor first.) Without the onion domain access leaks, we would not have had a reason to suspect that the existing Tor connections are related to malware. An approach relying on IP or domain blacklisting is irrelevant as the destination IP or domain appears as a benign entry guard IP for both benign and malicious Tor connections. Even if malware establishes multiple TCP streams, Tor often multiplexes them in a single circuit, which is in turn multiplexed with other circuits in one TCP connection, so their frequency is very unlikely to raise any alarms. Finally, approaches relying on DNS resolution (or records) features alone are not expected to be useful, as Tor usually resolves DNS within the network. These observations motivated

[1] As a result of not using an OP, this service does not provide user anonymity, and onion address search queries may leave traces in users' DNS logs.

[2] Those are: 57g7spgrzlojinas.onion, 76jdd2ir2embyv47.onion, cwwnhwhlz52maqm7.onion, gx7ekbenv2riucmf.onion, sqjolphimrr7jqw6.onion, Xxlvbrloxvriy2c5.onion, and cwwnhwhlz52ma.onion

us to explore approaches to identify Tor-based malware from the traffic and the logs.

Identifying such covert C&C connections would also be challenging for prior work. Table 1 compares our work with other related C&C or malicious traffic detection solutions. While valuable, these works have limitations that make them less ideal for our purposes. Their design is customized to capture specific malware behavior that may not be exhibited by stealthy Tor variants. Indeed, the detection approach used in some of these works is limited to identifying specific connection patterns in unencrypted botnet traffic based on C&C protocols such as IRC, HTTP, and P2P.

First, we require detection to be performed on the client network side. This allows a network admin, who has access to network and traffic logs, to identify local infections effectively. TorWard is a signature-based Deep Packet Inspection (DPI) solution designed to detect abuse traffic (spam, DoS, etc) at Tor exit routers. Jackstraws, on the other hand, detects traffic at the client side; however, it relies on host-based activities such as network system calls to generate behavior graphs, which can be more computationally expensive than traffic analysis. Furthermore, many organizations do not have access to DPI data or do not have access to the vantage point (exit node) to identify Tor malware.

Second, while some of these works perform network-based detection at the client side, they are not designed to detect stealthy Tor-based variants, nor can they differentiate between benign and malicious Tor connections. Botminer and BOTection, for example, rely on traces of coordinated bots and their anomalous bursty behavior (such as scanning behavior within local networks to be able to propagate). Single isolated Tor-based malware infections can evade detection with such systems.

To summarize, prior systems require network data that is dependent (coordinated) between multiple clients and servers [22, 35], whereas our work classifies Tor connections independently. Also, our goal is to differentiate between benign and malicious Tor connections, which is more specialized. As we have different assumptions, input requirements, and objectives, we do not compare our work empirically with previous work. However, we compare our classifier performance against previous traffic analysis Deep Learning approaches [26, 61] in Section 5.

2.3 Ethical and safety considerations

We follow all possible precautions to ensure safe malware research and compliance with Tor's safety policy. We conduct our experiments on Crowdstrike's Falcon online sandbox environment, which is widely used by researchers and the industry to conduct malware analysis research and investigation. While each run allows malware binaries to establish connections, each run is limited to six minutes.

Since we are performing traffic analysis on Tor connections, it is important to ensure we are not compromising users' traffic or privacy. Indeed, our data collection and analysis are performed on our data, and we do not touch other users' data. Admittedly, running Tor-based binaries creates malware connections through the Tor network; however, this does not affect its users as their circuits are isolated, and ORs only relay the corresponding cells to the destinations. Again, this is done during a limited time window that is generally acceptable in malware analysis research. Furthermore, our experiments do not overload the network. (Figure 4 in Section 3.3.2 shows the data exchanged in our connections.)

With the hundreds of Tor-based malware variants launched in the wild, devising techniques to eliminate such threats is paramount. We argue that the benefits of such research outweigh the limited risks we discussed above. In the spirit of open source and sharing, we share our traffic datasets, scripts, and models online [3]. We will share the malware binaries hashes with the research community upon request and consultation with the Tor project.

3 DATASET COLLECTION

3.1 Tor-based malware binaries

VirusTotal. We source our malware corpus from VirusTotal (VT), a threat intelligence sharing platform used by thousands of security researchers and hundreds of security companies. The VT platform aggregates detection results for over 72 different AntiVirus (AV) engines and reports their labels. On average, VT receives 1.8M unique files daily, of which 400K are detected as malicious, and of those, about 150K are windows malware [24]. As part of its threat intelligence, VT provides the number of AV engines that mark the file as malicious. A file object in VT contains several useful attributes including capabilities and threat category tags that provide a representative characterization of a file's capabilities and behavior (ransomware, spyware, etc) based on static and dynamic analysis tools.

Tor binary collection. We utilize VT by carefully inspecting behavior reports of various Tor binaries, we use the following criteria to search the VT platform for Tor-based malware:

(1) Search for keyword "tor". Tor usage is often highlighted in the reports by IDS analysis (e.g. "ET POLICY TOR Consensus Data Requested", "ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 313").
(2) Search for keyword "onion". This allowed us to find binaries that attempt to connect to onion domains (ending in .onion)
(3) Search for keywords related to the Tor process such as "tor.exe", "consensus" or "torrc". Although some malware binaries change the name of the Tor process to avoid detection, we observed the torrc configuration file is still detectable.

Tor usage verification. We downloaded the PCAPs of the collected binaries and examined their traffic for Tor connections. We use Zeek [21] to generate logs that organize PCAP information based on application protocols. We use `conn.log`, which contains the connection details of TCP traffic, and `ssl.log`, which contains the SSL handshake certificate details for the verification step. Tor traffic utilizes TCP; therefore, we inspect the `conn.log` files of all binaries to identify a connection with `dstip` = Tor router IP and `dstport` = Tor router port. As for SSL logs of identified Tor connections, we use them to verify router certificates. Tor router certificates have peculiar SSL CommonName (CName) or servername field values containing random strings ending with .com or .net (e.g. www.q7edykf6lbgsuh.net [56]).

Identification of Tor-based malware binaries. We used the VT malicious score to ensure that the binaries we are collecting are

[3] https://github.com/malfp/tormalwarefp

indeed malicious. The score is the number of VT engines that detect the binary as malicious. We collected 559 Tor-based binaries which we filtered down to 523 malicious Tor-based binaries. We removed 36 binaries that had a score of 0. The remaining set of 523 binaries had a median VT score of 44 (q1 = 37 and q3 = 51), indicating that they were highly likely to be malicious.

Malware traffic collection. We analyze the Tor-based malware binaries on Falcon sandbox [4], a public malware analysis service from Hybrid analysis. The sandbox executes the binaries and then generates PCAPs and analysis reports. At the time of writing, a free verified account for the sandbox allowed up to 100 submissions/day with six minute execution time limit per binary.

We implemented the above binary collection, verification, and traffic collection steps in a pipeline that ran daily between June and September 2021. Our binary collection module queries fresh (submitted within the last 24 hours) Tor-based binaries on a daily basis against the full VT database of malware reports to ensure that fresh active binaries are collected. Next, our traffic collection module submits those binaries to the deployment sandbox. The traffic instances are then collected over several days until we have submitted each binary 60 times. This way, we gathered the same number of PCAPs per binary over the data collection period. Running both modules simultaneously ensures that we are capturing fresh samples that are more likely to generate traffic and avoid stale binaries (outdated binaries may not generate bidirectional traffic.)

Challenges. Analyzing malware in a sandbox system presents several challenges. First, some binaries attempt to evade sandbox analysis by sleeping, terminating, or alternating behavior when a sandbox environment is detected. We identified these cases when the sandbox analysis reports lacked network traffic. Therefore, different binaries generated different fractions of Tor traffic despite the equal number of submissions for each binary.

Second, Tor-based malware network activities are highly variable and unpredictable. For example, malware sometimes creates scheduled tasks during which they establish Tor connections. Such connections may not be visible during a sample run because the sought-after Tor connections are scheduled outside of the limited sandbox run window or because the C&C may not be online or alive during the run time of the corresponding binary. Although Falcon sandbox offers some features that address evasion tactics by malware, we do not consider malware samples that require system restart because the free version of Falcon does not support restart during analysis. We also exclude malware binaries that do not initiate Tor connections within the analysis window. For future work, we recommend a more resilient sandbox technology to potentially improve detection.

Generated malware traffic. In total, we obtained 5,984 PCAPs from 362 active malware binaries (out of the collected 523 binaries) which successfully generated traffic during their sandbox deployment window. We summarize the numbers of PCAPs per binary and derive experimental datasets based on this data in Section 4.2. As we discussed, it is difficult in this domain to obtain a larger set of Tor malware binaries that are active within the limited sandbox deployment window. One downside to our small malware corpus size is that it may affect the generalizability of our observations and results. We note however that our malware corpus contains a diverse set of malware families and classes (Section 3.3.1). Same-class binaries share patterns allowing our features to label them with their behavior class, which aids the detection of newer novel variants. In practice, machine learning detection models require retraining with newer samples periodically to be able to predict emerging variants and stay sharp against concept drifts.

3.2 Benign traffic

For the classifier to work effectively in the real world, we ensure it is trained on benign traffic covering various client applications. This is done by using (1) our own generated traffic of various loads, and (2) an independently collected public dataset of benign traffic consisting of various applications.

Generated browsing traffic. Tor is mostly intended for browsing. We follow the footsteps of previous works in the traffic analysis and WF literature in simulating web browsing clients over Tor using multiple profiles and usage loads. We use Tor Browser 10.5.2 in our scripts and our clients visit the latest published list of top Alexa domains [19]. For hidden services traffic, we use Ahmia [7], an onion service search engine and directory, to find 1000 onion-sites we obtained from keyword searches for common categories including bitcoin, cryptocurrency, social media, news, and technology. We ensure the liveliness of these onion domains and that their accessibility is unaffected by the deprecation of onion v2 sites during the transition to v3 in October 2021 as announced by the Tor community [5]. We confirm that all onion domains in our list return an HTTP 200 response before executing our collection scripts.

We introduce three client profiles simulating different frequencies of accessing sites simultaneously using multiple tabs and windows. We do this using random time gaps between site accesses, similar to the negative, positive, and zero-time separated page access methodology adopted in some Tor-based traffic analysis or WF works [67, 69]. These user profiles are integral to the quality of our benign ground truth, as the frequency and number of sites accessed affect the number of web pages loaded during a browsing session. This, as a result, determines the amount of data exchanged, packet timing, size, and direction of packets collected during a browsing session simulation. In particular, we define the following user profiles:

(1) **Light.** Sites (index page only) are loaded with a positive time gap ranging from anywhere between 20 seconds to 10 minutes. Here, we assume that a user may spend longer periods of time before visiting another site.
(2) **Medium.** A more active user is simulated with sites accessed with a gap that ranges between 20 seconds to 1.5 minutes. This may result in partial site loads running in the background while multiple other sites are loaded simultaneously. This is similar to a scenario where a user may open Youtube to play music and switches within a few seconds to read news on CNN followed by visiting a cryptocurrency onion-site, for example. At the network level, this may result in more than one active Tor connection.
(3) **Heavy.** This is similar to the medium profile above with an even smaller time gap between site loads. Here, pages are loaded within 2 to 20 seconds.

Using domain lists and different user profiles, we develop Python scripts to access sites in the Tor Browser using the `Tbselenium` driver library. We create scripts with the three profiles described above (light, medium, and heavy) accessing only Alexa sites (popular and less popular), only onionsites, or a mix of both. We test these scripts on a virtual machine running Windows 32-bit operating system. We bundle the Python script, domain list, and Tor browser to create a single windows PE32 executable to run on the sandbox using Hybrid Analysis web API.

Collected data. We execute our bundled executables in the sandbox over two collection periods totaling three weeks. The first two-week collection period uses Alexa top 1K domains, whereas the second collection period uses less popular Alexa domains (Alexa 1M excluding Alexa 1K). We combine 3,540 and 1,045 PCAPs from the first and second collection periods, respectively, to diversify, generalize, and enrich our benign dataset with domains from different levels of popularity and provisioning. Furthermore, we intentionally construct the dataset to include substantially more traffic from popular domains as we expect those domains to be visited more often in practice. The 4,585 PCAPs have a breakdown of 38% light, 40% medium, and 22% heavy browsing traffic.

ISCXTor2016 [2] We combine our benign Tor traffic with a public dataset collected by Lashkari et al. [2, 41], which has been used previously to classify Tor traffic in a network [39, 44, 57]. The data is auto-generated in a lab setup consisting of a workstation based on Whonix, a Tor-based Linux OS simulating users accessing 18 applications including Skype, Gmail, Spotify, Youtube, and Vimeo. For our purposes, we only use raw PCAPs corresponding to Tor traffic from six categories: browsing, chat, mail, FTP, audio, and video streaming. Because our malware PCAPs run for six minutes in the sandbox, we used the first six minutes of all the available 30 PCAPs in this dataset.

3.3 Malware binary & traffic characteristics

In this section, we study and describe the characteristics of the *active* Tor binaries and traffic we collected in Sections 3.1 and 3.2.

3.3.1 Families and classes. We use AVclass [60] and AVclass2 [59], which are automatic malware tagging tools to categorize binaries into malware families and classes. AVClass uses a majority voting approach with advanced label and text normalization techniques for binary categorization. Family labels are common names that security companies use to identify specific malware threats. Class labels are generic labels that define the malware type, such as worms, trojans, hacking tools, ransomware, and others. Family labels are more specific, while class labels are more generic and are assigned by AV engines that may not have coverage for specific families or they fail to correctly label them [47, 48].

As input, they take VT-generated reports and output a summary for each binary hash containing its AV score, binary class (e.g., worm, ransomware, grayware, etc), and its most likely malware family (e.g., zeus, agentb, wannacry, etc). In this analysis, we use the 362 active (out of 523) malware binaries that generated Tor traffic. All binaries (active and inactive) belong to 80 families. Removing the inactive ones resulted in excluding 23 families, for which we had a limited number of inactive variants (these are listed in Appendix C, Table 9). Figure 2 depicts the distribution of binaries based on

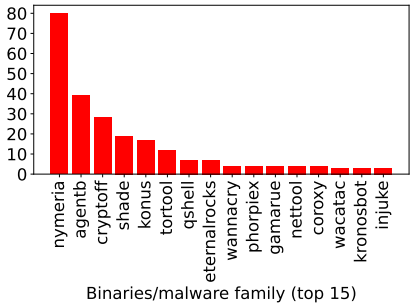

Figure 2: Malware family distribution in active binaries

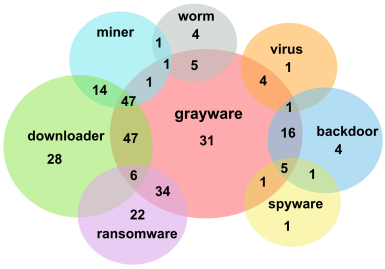

Figure 3: Malware classes in active binaries

malware families. The top three families are Nymeria, Agentb, and Cryptoff with 80, 39, and 28 binaries, respectively. Other malware family types covered in the dataset include Tortool, WannaCry, Eternalrocks, weed, Enigmaprotector. There were 75 binaries that AVclass categorized as "singleton" meaning their families were unknown.

A binary can have one or more classes based on its behavior. Active binaries in our dataset are characterized by 10 unique classes. We observe that the majority of binaries belong to the grayware [14] class which is a generic categorization used for malware and usually includes other classes like spyware [17] or adware [20]. Figure 3 shows the malware class distribution of all our active binaries.[4] Indeed, various binaries show multiple classes of behavior. For example, 47 binaries are labelled as "grayware", "miner" and "downloader" and 34 labeled as "grayware" and "ransomware". The observations above show that our active binaries are diverse as they come from different families. Second, Tor is clearly popular for various behavior classes and families of malware.

3.3.2 Characterizing Tor malware connections. Our active malware binaries produce a total of 30,592 Tor connections after excluding idle connections.[5] We experimentally determine that idle connections have less than 32 cells, including initial circuit construction cells. The active connections represent a successfully established Tor circuit with an end server. On average, binaries in our dataset hold a connection for at least 19 seconds with an average of 181 TLS packets sent and 393 received in all connections. Figure 4 shows the CDF of data sent and received by binaries in Megabytes (MB). As can be seen in the figure, data received in malware connections is

[4]For concise presentation, smaller intersections of binary classes were omitted.
[5]Tor creates multiple spare circuits for performance reasons that are sometimes never used.

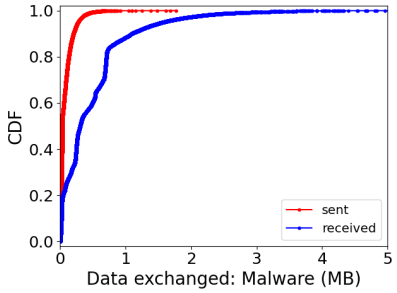

Figure 4: CDF of data exchanged in active malware binaries

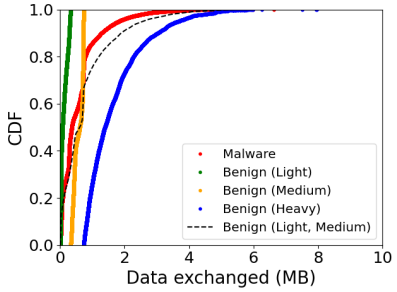

Figure 5: CDF of total data exchanged in malware and benign Tor connections

comparatively more than data sent. Indeed, 72% of Tor connections send up to 0.1 MB and receive up to 0.68 MB.

As for the benign traffic generated in the same sandbox, we obtain a total of 32,275 Tor connections (after excluding idle connections). Connections in our benign traffic last at least 38 seconds on average with an average of 259 and 574 TLS packets sent and received in each connection, respectively. Figure 5 depicts the CDF of the total data exchanged in each Tor connection for our malware and benign datasets. The three benign profiles generated connections with a wide range of traffic loads as intended. In fact, the distribution of the data exchanged in malware Tor connections appears to be particularly similar (and the distributions even intersect), to the light and medium benign profiles, for roughly 70% of the connections, exchanging up to 0.7 MB of data. The similarity in benign and malware traffic makes for representative experimental datasets that reflect real traffic captures with higher chances of malware traffic blending into benign.

4 FEATURES AND DATA PREPROCESSING

4.1 Classification features

Connection-level features. A large body of related works in the WF attack literature is based on extracting Tor traffic features for precise webpage classification. The precision achieved in these attacks mainly relies on time, direction, order, and concentration of packets in Tor connections. These features characterize a unique website fingerprint. Although it is not our goal to precisely expose the sites being accessed, these features are generalized to identify useful traffic patterns. For our work, we intend to capture malware communication fingerprints from these features such as C&C beacon communication, malware payload download attempts, access to specific sites, and use of the Tor browser by malware.

Table 2: Host-level features extracted using all Tor connections per PCAP (including idle connections)

Category	Category-based Features
Duration	Average/shortest/longest duration connection No. of short duration connections (<= 1 minute) Average duration between each Tor conn
Data	Mean/median/mode of total data exchanged Mean/median/mode of total data sent/received Mean/median/mode of total packets sent/received
Port	No. of unique DST ports used across connections Most frequent DST port used across Tor conns No. of non-standard DST ports seen Most frequent non-standard DST port
Connection	No. of connections seen (per host or PCAP) No. of failed or rejected attempts No. of connections per second No. of failed attempts per second
DNS	No. of DNS queries rcode_name: REFUSED No. of DNS queries rcode_name: SERVFAIL No. of URLs seen using "consensus" keyword No. of URLs with "\tor" keyword No. of DNS queries rcode_name: NXDOMAINS Total no. of leaked onion domains No. of unique onion domains leaked No. of 'rejected' onion domain queries

To this end, we use the features proposed by Hayes *et al.* [36]. This set consists of 150 WF features, and in general, the set comprises a superset of the features used in prior attacks. Hayes *et al.* justifies the effectiveness of the feature set in extracting the most information from encrypted traffic, which exactly matches our goals. We extract WF features for the top three active Tor connections per PCAP. Note that a Tor client typically chooses three entry guards for months and establishes TLS-encrypted TCP connections to them. All circuits are then multiplexed in those three connections. Indeed, we observed in traffic PCAPs that the top three are sufficient to get all active data-exchanging Tor connections, which is precisely what we need for traffic analysis.

Host-level features. In addition to WF features, we introduce our set of novel features to capture malware behavior that may be exposed by analyzing *all* Tor connections initiated by a host, including failed and less active connections at a host level (or PCAP). This set consists of 40 generic features in total with 22 novel features that capture Tor-based malware activity as listed in Table 2. Below we elaborate on our observations and intuitions about these features.

Malware attempts to connect to C&C are better captured by looking at the number of short-lived Tor connections seen on a host in the event of failed attempts, the frequency with which these attempts are made, and the corresponding DNS activity. Our duration features include the average duration of all Tor connections and other related statistics such as the minimum and maximum

durations of connections and the number of short duration (<= 1 minute) connections seen in a PCAP. To capture the frequency of connections, we formulate features such as the number of Tor connections per PCAP, number of failed connection attempts, number of connections per second, and number of failed attempts per second. We used the average time gap (in seconds) between each Tor connection as a feature to capture unusual connection patterns.

It is possible that malware may fail to use Tor successfully to contact its destination. In such cases, it may utilize other methods to access its hidden service such as via Tor2Web. Doing so can lead to onion domain leaks in the DNS activity of an infected host. Moreover, some malware families are known to contact kill switch domains (onion sites hard coded in the binary) which signal the malware to execute certain operations. WannaCry, for instance, uses a failed response from its kill switch domain as a signal to spread to other machines in the network [3]. We capture these scenarios with DNS features such as the number of DNS queries with a 'REFUSED,' 'SERVFAIL,' or 'NXDOMAIN' response, number of onion domain leaks, number of unique onion domains leaked and rejected onion domain queries.

We also observe that some binaries bundled with unofficial or modified versions of Tor (such as mini Tor [52]) try to evade detection by contacting private servers for router consensus information.[6] These may often contain stale router information, in which case we observe the usage of stale Tor ports in malware traffic. This in turn causes the malware to attempt to establish multiple Tor connections. We use features such as the number of destination ports seen in a PCAP across all Tor connections, the number of unique DST ports seen[7], most frequently used DST ports, and the number of HTTP URLs accessing sites with keyword 'consensus' or '\tor' to capture such activities. Finally, we use statistical features such as the mean, median, and mode of the total data sent and received and packets exchanged across all Tor connections.

Differences between benign and malicious Tor traffic arise from (1) server traffic fingerprints (patterns, burstiness, lifetime of connections, frequency, etc) and (2) client-side anomalies. Connection-level features effectively fingerprint specific servers and their pages. Host-level features are computed at the PCAP-level and capture client-side anomalies such as short-lived, and sometimes failed connections due to trying stale routers (IPs no longer in Tor consensus).

4.2 Extracting Tor connections

We extract Tor connections and verify them in a way similar to the traffic verification approach we described in Section 3.1.[8] In addition to using `conn.log` and `ssl.log` for extracting connections, we also use `dns.log` and `http.log` fields for feature engineering. Once Tor connections are extracted from PCAPs and verified in the pipeline, we parse the corresponding connection packets for Tor cells. Tor embeds application data into Tor cells of size 514 bytes [31]. We follow the standard parsing methodology used by previous Tor traffic analysis work (such as Wang *et al.* [66]) as follows.

A TLS packet consists of one or more Tor cell units. To derive the number of cells in a TLS stream, we consider the length of the TLS application record, round it to the closest multiple of 514, and divide it by 514 (the size of one Tor cell unit). For instance, a TLS record of length 1,088 results in two cells. If this record belongs to an incoming packet, it is marked as negative, otherwise marked as positive. The signs indicate the direction of flow. Each PCAP may contain one or more Tor connections, each of which is parsed, and stored in cell files as described above using Python's `dpkt` library [16]. Each line in a cell file corresponds to a single cell with its time and direction information. The time is calculated relative to the timestamp of the first packet of a Tor connection. By the end, we have a cell file, which consists of cells corresponding to the top three most active Tor connections for each PCAP (in terms of the number of cells). Note that some PCAPs may have fewer than three active Tor connections.

In total, we obtain 13,214 Tor connections from benign PCAPs (refer Section 3.2 for benign data collection methodology). For malware traffic, recall from Section 3.1 that we have 5,984 malware PCAPs generated by 362 binaries. From these, we derive four experimental datasets (similar to the n-shot learning approach in *Triplet-Fingerprinting* [62]). Each dataset is denoted as *DN* where *N* is the number of PCAPs selected uniformly at random per binary. Table 3 summarizes these datasets with their respective notations, the number of binaries in the dataset, PCAP instances with malware traffic, and total number of classifier instances (Tor connections) in the dataset. D5, for example, consists of traffic from 157 malware binaries, each of which has five PCAPs, all containing malware traffic. Finally, extracting the top three most active Tor connections from each of these five PCAPs results in 2,027 classifier instances in this dataset.

Table 3: Experimental malware traffic datasets

Malware dataset	Number of binaries	PCAPs per binary	Total classifier instances
D5	157	5	2,027
D10	130	10	3,657
D20	107	20	6,135
D30	62	30	5,342

5 BINARY CLASSIFICATION: CAN WE IDENTIFY MALWARE CONNECTIONS?

For evaluation, we use the following metrics: Precision, Recall, and the False Positive Rate (FPR). Precision measures the ratio of correct positive class predictions among all positive predictions reported by the classifier. This defines how reliable model predictions are if they were to be used in a real system taking into account the number of false alarms. Recall measures the classifier's ability to identify positive class samples from all actual positive samples. A classifier with very low recall would miss most malware instances classifying them as benign, making false negatives high in this case. Finally, the FPR is the most important metric, and our goal is to keep it as low as possible so as not to overwhelm network admin users with the rate of false alarms.

[6] Tor ORs and OPs download the consensus information from the Tor directory servers.
[7] Tor and Tor Browser use standard ports: 9001, 9010, 443, 9050, 9150, 443, 80, 8080
[8] Recall that we previously used VT PCAPs to verify that the binaries we find using our search heuristics do indeed use Tor.

5.1 AutoGluon

For binary classification experiments, we utilize AutoGluon [32], an Automated Machine Learning (AutoML) tool developed by Amazon. AutoGluon trains at least eight base machine learning and two neural network models on raw tabular data.

Training and best performing model. The tool trains each model iteratively using random stratified splits of data for training and validation with different hyperparameters in each run. During training, it tries to optimize model performances on validation data based on an `evaluation_metric`. By default, it is set to the *accuracy* metric. For our work, we change this to *balanced_accuracy*. We found it to be the best metric to strike a good balance between precision and recall while maintaining a low FPR. At the end of the training phase, Autogluon reports the best-performing model based on the chosen metric. Additionally, it summarizes the performance scores of all trained models with the best hyperparameter setting on the validation and test datasets (if provided by the user).

Base models, stacking, and ensembling. AutoGluon uses Light Gradient Boosting Machine (LightGBM), CatBoost, XGBoost, Random Forests, ExtraTrees, kNN, Logistic Regression, and Tabular Neural Network models along with stacking and bagged ensembles of each (refer to the reference documentation [9] for definitions of each model type). Additionally, it offers multi-layer stack ensembling called `WeightedEnsemble`, where a stacker model takes the predictions of base models along with base features. Predictions of stacker models are used as features for models at higher levels. In the final layer, a weighted ensemble of stacker model predictions is created using greedy forward ensemble selection. Predictions of a model are weighted higher if it improves the evaluation metric on validation data.

5.2 Performance evaluation: All datasets

Since we have four different datasets with different numbers of traffic instances per binary, we first seek to understand if *increasing the number of PCAPs or training instances per binary impacts the performance of the binary classification problem.*

Training. We train autogluon models using our D5, D10, D20, and D30 datasets (summarized in Table 3). We carry out each DN experiment as follows. First, we split PCAPs into separate training (70%) and testing (30%) datasets.[9] Next, from each PCAP, we extract its top three active Tor connections, and compute their connection- and host-level features to create the corresponding classification instances. For each $PCAP_i$, we have at most three classification instances CI_{iA}, CI_{iB}, and CI_{iC} defined as follows:

$$CI_{iA} \rightarrow (cf_{iA1}, cf_{iA2}, \ldots, cf_{iAn}, hf_{i1}, hf_{i2}, \ldots, hf_{ik}, CLASS)$$
$$CI_{iB} \rightarrow (cf_{iB1}, cf_{iB2}, \ldots, cf_{iBn}, hf_{i1}, hf_{i2}, \ldots, hf_{ik}, CLASS)$$
$$CI_{iC} \rightarrow (cf_{iC1}, cf_{iC2}, \ldots, cf_{iCn}, hf_{i1}, hf_{i2}, \ldots, hf_{ik}, CLASS)$$

Variables cf and hf denote connection and host features, respectively. For every extracted Tor connection, there is a corresponding classification instance that combines both connection- and host-level features. Connection-level features characterize the corresponding TCP connection, whereas the host-level features characterize the host/PCAP. Note that all host-level features from the same PCAP are identical as they represent global host variables across all connections (e.g. total number of Tor connections in a PCAP, total data exchanged across all Tor connections). For example, in D5, connection-level features are derived from 2,027 TCP connections (Table 3), but more than 47,000 connections are used to compute its host-level features. For benign traffic, we get a total of 13,214 classifier instances from 4,615 PCAPs. Note that the models are trained on imbalanced data as we have more benign connections than malware in each dataset. This configuration is intentional as it reflects the expected real-world scenario where benign browsing traffic is more widespread than malware traffic.

[9]We also discuss a different splitting approach in Appendix B.

Table 4: Performance comparison of different datasets

Dataset	Precision(%)	Recall(%)	FPR(%)	AUC(%)
D5	93.3	81.6	0.88	98.56
D10	95.3	83.0	1.06	97.16
D20	96.2	89.6	1.55	97.76
D30	95.7	88.8	1.52	97.42

Test results. Table 4 compares the precision, recall, FPR, and AUC for all datasets. In general, malware connections are predicted with high precision, recall, AUC, and very low FPR for all datasets. Increasing the number of malware traffic instances per binary used to train models at the expense of a reduction in the total number of unique binaries does not significantly impact performance. We observe a noticeable improvement in recall and precision by increasing the number of instances for each binary up to D20, after which this gain slightly drops for D30 (possibly because it covers a smaller number of unique binaries). However, FPR appears to be increasing as the number of unique binaries decreases. With D5, FPR is 0.88% and it gradually increases to 1.55% and 1.52% for D20 and D30, respectively. Following our observations, we choose **D5** as the main dataset for further experiments as it consists of traffic from the largest number of binaries (157) compared to the other datasets and achieves the lowest FPR and highest AUC indicating a superior ability in separating between the classes.

5.3 Impact of features

Experiments. We formulate binary classification experiments using D5 (See Table 3 for a summary of the dataset). The goal here is to investigate the impact of host-level and connection-level features and compare the performance of Autogluon models, which we treat as a black box, and other recent Convolutional Neural Networks (CNN) traffic analysis proposals [26, 61]. For this, we use our 13,214 benign and 2,027 malware classification instances in D5. These are used to train all machine learning and deep learning models in Autogluon (Section 5.1) along with Var-CNN [26] and DF [61], which are deep learning models used to carry out WF against Tor.

Additionally, we evaluate Autogluon models using host-only features with 785 malware and 4,615 benign classifier instances. Recall that the host-level features are derived globally per PCAP using all Tor connections, hence the number of classification instances reduces in this case compared to when using the top three active Tor connections per PCAP. For training and testing the classifiers, we use the standard 70-30 (train-test) split on PCAPs in all cases.

Table 5: Binary classification performance on D5 using different models and features

Experiment	Model	Precision(%)	Recall(%)	FPR(%)	AUC(%)	Feature set
E1	XGBoost	86.13	63.37	1.53	93.62	Connection-level
E2	LightGBM	90.96	76.34	1.45	96.91	Host-level
E3	LightGBM	93.33	81.60	0.88	98.56	Connection & host-level
E4	DF	75.51	60.76	3	91.24	Connection-level
E5	Var-CNN	85.65	35.59	0.9	86.27	Connection-level
E6	Var-CNN	91.48	55.13	0.78	89.23	Connection & host-level

Results. Table 5 summarizes the experiments with the features and models utilized in each, along with the precision, recall, FPR, and AUC results. Note that for E1, E2, and E3, the model listed is the best performing model as ranked by Autogluon. We observe that for experiments E1 to E3 using Autogluon, LightGBM and XGBoost outperform all other models including FastAI and MXNet library-based neural network models for tabular data. This is not surprising as LightGBM and XGBoost are based on decision trees, which are known to perform well in various classification problems.

While connection-level features from Hayes *et al.* in E1 perform well, we observe that our novel host-level features in E2, outperform them in all summarized accuracy metrics. The model in E2 detects more malicious connections, compared to its connection-level counterpart in E1, as evidenced by the significant improvement (roughly 20.5%) in recall achieved, all the while maintaining improved precision and FPR. The AUC score is also higher in E2 for host-level features compared to connection-level features in E1. These results indicate that high-level information extracted using all Tor connections in a PCAP can be more effective in capturing and classifying malware behavior than when activity in only the most active connections is considered. This is handy since most enterprise logs have these readily captured as opposed to packet capture logs needed for connection-level features.

That said, we obtain the best performance when both feature sets are combined. LightGBM performs best with a high precision of 93.3%, 81% recall, and a very low FPR of 0.88%.

State-of-the-art DL performance. In E4 and E5, we evaluate the performance of CNN models proposed in DF and Var-CNN. We used the recommended parameters with different epochs. Note that in DF, the authors only use automatically extracted features from the CNN model, whereas in Var-CNN, a semi-automated feature input is used for the CNN-based ResNet-18. The model is fed seven additional features along with the auto-extracted features from the raw data. These seven features are the total incoming and outgoing cells, the ratio of in/outgoing cells to the total number of cells, the average number of seconds between each outgoing cell, and the total transmission time.

The results of applying these CNN models on our data show acceptable precision performance (75.51% in E4 and 85.65% in E5) with low FPR (3% in E4 and 0.9% in E5) at the expense of poor recall performance. The models can predict very few malware-related connections with low FPR but miss detecting a substantial number of malware instances. Most malware connections are labeled benign by the models. However, when we add our host-level features along with Var-CNN features, we observe some improvement in recall. The model in E6 achieves a precision of 91.48%, 0.78% FPR, and 55.13% recall. One reason behind the significantly low recall in these models may be attributed to the dataset size. For the automatic feature extraction to work effectively, we would require to feed in a significantly larger number of malware-related Tor connections. Given the malware binary and traffic collection challenges, getting sufficient data for training is infeasible. These results clearly indicate the superiority of LightGBM in effectively classifying malware connections over state-of-the-art CNNs when restricted to a small training dataset.

Feature importance. Features in Autogluon are ranked based on an "importance" score calculated using the permutation importance. The score measures the performance drop resulting from shuffling values of the features and evaluating the model on a modified copy of the data. A high score signifies higher importance of the feature in model performance [10]. The most important connection-level features for the best performing model in experiments E1 and E3 fall into the following categories:

- **Cell inter-arrival time features:** These include statistical measurements of cell interarrival times such as the maximum interarrival time between all cells (in and out), between all incoming cells, and between all outgoing cells.
- **Cell concentration features:** This is the count of incoming and outgoing cells in the first thirty cells in a Tor connection.
- **Outgoing cell features:** These include the standard deviation of the total number of outgoing cells that appeared before each successive outgoing cell in the sequence and the percentage of outgoing cells of the total cells in a connection.
- **Rate of cell features:** This is the alternating number of cells per second, where a list of the total number of cells per second is split into 20 equally-sized subsets and each subset is aggregated.

The host-level features that lead to noticeable precision and recall improvements in E2 and E3 result from the top five features listed in Table 2 by their order of importance. The top three features use the duration of connections such as the average, shortest, and longest duration of connections. As intended, these features leverage peculiar differences in the duration of malware and benign traffic, which is not captured at the connection level. Other top-performing features include the number of unique DST ports used across Tor connections, which reflect distinguishable patterns in the variety of ports used in malware compared to benign traffic. Other features that make up the top 10 ranked include statistical mean, median, and mode of data sent and received in all Tor connections.

Statistical significance. We use the 'p-value' scores of the top 10 features to quantify their usefulness to the classifier predictions.

The 'p-value' score in Autogluon is derived based on a statistical t-test of the null hypothesis on the importance score (importance = 0). A p-value close to zero indicates a very low possibility of the feature being useless to the model [8]. We note that the average p-value of the top 10 features for the best-performing model in E3 is 0.093. Note that we obtain nine host-level and one connection-level feature in the top 10 rankings. The low p-value scores of these features indicate that these are highly useful to the model predictions.

In summary, we observe that our global host-level features outperform connection-level features, so they can be used solo in practice when traffic is not available to extract connection-level features. However, we achieve the highest performance when we combine both host- and connection-level features.

6 CAN WE IDENTIFY THE MALWARE CLASS?

An essential usability feature for real-world applications of our proposition is in identifying the class of the Tor malware traffic detected on the network. This translates to a classical multi-label machine learning classification problem as each malware instance could belong to one or more classes. For example, a WannaCry malware binary can be categorized as a "ransomware" and a "worm". To achieve this goal, we evaluated a random forest model with 3 multi-label classification techniques namely, Binary Relevance (BR) [11], Classifier Chains (CC) [12], and Label Powerset (LP) [15]. We utilize Python's `scikit-learn` machine learning library for the model and `scikit-multilearn` library for the multi-label techniques. For training and testing, we use a 70-30 (train-test) split across all PCAPs.

We train the model using these techniques with D5. We label the binaries in D5 with their respective class labels derived from AVCLASS2 (described in Section 3.3.1). The binaries in D5 fall into the following 9 classes: 'grayware'(94), 'downloader'(88), 'ransomware'(26), 'miner'(31), 'worm'(6), 'keylogger'(1), 'spyware'(3), 'backdoor'(4), 'virus'(4) and a 'singleton'(12) category synonymous to 'unknown' label, where the number in parenthesis corresponds to the number of binaries that belong to the class type. All binaries in D5 have two labels on average. The class labels assigned per binary range from a minimum of one label to a maximum of four.

As an evaluation metric, we use standard multi-label classification metrics, which are namely, the hamming loss, and the micro average precision and recall instead of accuracy metrics. Using accuracy can be misleading because it measures the fraction of correct predictions which requires predicted labels to exactly match all the true labels for each Tor connection. Hence, it does not portray classifier potential in predicting partially correct labels. Hamming loss takes this into account by counting incorrect labels in predicted vs actual labels and averages this distance over all samples. An ideal classifier with no incorrectly predicted labels has a hamming loss of 0. Furthermore, given the class imbalance in our dataset, we use micro-average precision and recall over the macro-average metric. The former aggregates True Positives, False Positives, True Negatives, and False Negatives across all classes to calculate precision and recall. Macro-average, on the other hand, calculates the precision and recall for each class individually before averaging.

Table 6: Malware class label prediction performance

Classification technique	Hamming loss	Micro-average precision(%)	Micro-average recall(%)
Binary Relevance	0.1	68.12	70.77
Classifier Chains	0.1	67.77	71.05
Label Powerset	0.1	66.81	72.37

Table 6 summarizes the best model performances with different multi-label classification techniques. Recall that the model is trained on 70% of the malware PCAPs in D5 (1,423 connections) and tested on 30% PCAPs (604 connections). From the results obtained, we observe that all techniques have a small number of misidentified class labels with a hamming loss of 0.1. The model trained with LP predicts the most number of correct label combinations with 72.37% recall, the highest of all techniques. In summary, our features and models can predict the behavior class of malware from their traffic.

7 ZERO-DAY EXPERIMENTS

In this section, we evaluate how our best performing models (discussed in Section 5.3 and Section 6) perform in the face of new malware binaries never used in the training process. We call this experiment *zero-day* test as we fetch fresh new Tor-based binaries from VT using the same methodology described in Section 3.1. We obtain 57 binaries on a single day in December 2021. We use these to collect malware traffic. We execute these binaries daily on the falcon sandbox for one week with an average of 400 submissions per day. Note that these malware binaries and their network traces are collected almost three months after the initial binaries were collected to build our binary classification model described in Section 5.

Upon inspecting the collected malware PCAPs for the new 57 binaries, we observe that only 26 binaries generated traffic resulting in a total of 192 PCAPs. This is not surprising given the challenges we faced in data collection as discussed in Section 3.1. For benign traffic, we execute our benign data collection scripts (described in Section 3.2) to generate a fresh test dataset that is disjoint from the previous datasets used. This resulted in browsing profiles of roughly 75% light and medium browsing, and 25% heavy browsing. Our scripts access either Alexa[10] domains, onion domains, or a mix of onion and Alexa domains during execution (6 minutes). In total, we obtain 1,010 PCAPs with benign Tor traffic. We process malware and benign PCAPs and extract cells from active Tor connections. For malware, we obtain two binaries that generate active Tor connections in some of their executions. These are:

B1: SHA-256: 413ebd37620cfcb229322a0f3217ae8a6a61163eb 73b14a30e3d8d5a68847f1b (7 active instances)

B2: SHA-256: 5e244bf3a2fe36942e9e001bbb6677f6c8e2dbbd1 d74f74a8d0cb9f78c9e4a57 (6 active instances)

B1 and B2 were uploaded and analyzed on VT on 2021-11-29 and 2021-11-14, respectively. According to AVCLASS2 labelling, they

[10]Roughly, 70% of Alexa accesses are for popular domains from Alexa top 1K, and 30% are from Alexa [1K,1M].

Table 7: Zero-day experiment results

Malware in Test	FPR	Precision	Recall
1%	1.1%	54.5%	100%
5%	0.7%	87.5%	100%
10%	1.0%	91.3%	100%
20%	1.2%	95.4%	100%

are both classified as `grayware` and `worm`. Upon further manual research, we discover that these binaries belong to a malware family known as EternalRocks [13] characterized as `ransomware, worm, downloader` by behavior. They are newer advanced successors of WannaCry. Variants of this malware family are designed to install and use the Tor browser stealthily to connect to their hidden C&C servers hosted on the Tor network. Having Tor connections with C&C traffic from such binaries is exactly the challenge we would like to use to test our trained model. This is also because these variants use the Tor browser, which is similar to our simulated benign user. Depending on the activity of this malware in the extracted Tor connections, there is a high chance that its traffic can appear as benign for our classifier.

The dataset used for this experiment consists of 42 Tor connections from B1 and B2 malware and 2,953 benign Tor connections derived from our browsing scripts. Recall that we extract the top three most active Tor connections for connection-level features and all Tor connections in a PCAP for host-level features as input to the classifier (described in Sections 4.1 and 4.2).

Binary classification. Our first goal using zero-day data is to identify malware Tor connections from benign successfully. We use the best performing LightGBM model with connection- and host-level features (E3 in Table 5). Note that the model was trained on D5 which does not contain binaries from the EternalRocks malware family used in this zero-day test. We evaluate our classifier under different situations by varying the proportion of malware and benign traffic. With this, we intend to simulate realistic scenarios where the number of malware Tor connections in a PCAP is expected to be extremely low relative to the benign connections.

We create four test scenarios using 42 malware connections and 2,953, 808, 370, and 170 benign connections. This corresponds to approximately 1%, 5%, 10%, and 20% malware connections for the total number of connections used in each scenario, respectively. Table 7 summarizes the results in all cases. The classifier can identify all malware connections (100% recall) with low FPR regardless of the proportion of malware traffic in the test set. In the 1% malware connections scenario, the classifier can achieve 1.1% FPR with 54.5% precision. As expected, precision degrades in this case due to the heavy class imbalance as the number of positive class test instances (42) is significantly lower than that of the negative class instances 2,953. Furthermore, even a small number of false positives (35 in this case) can take its toll on precision. Despite the low precision, the FPR is unscathed as the number of false positives is much lower relative to the total size of the negative class in the test. The FPR in all cases ranges between 0.7% to 1.2%.

Malware classes. Next, we turn our attention to the problem of predicting correct informative malware class labels for binaries B1 and B2 using our models. Having a model that can accurately predict the type of zeroday malware from Tor connections on an infected host serves as valuable information. Here, we use all three trained random forest models based on the different multi-label classification techniques we presented in Section 6. We use the model trained with connection- and host-level features. Note that the 42 instances of the binaries have the following true labels: `ransomware, grayware, worm,` and `downloader`.

Using the evaluation metrics used in Section 6, we find that the model trained with LP provides incremental improvements compared to that of BR and CC. The LP model has the highest recall with the lowest hamming loss (average number of incorrectly predicted labels). The model predicts at least one class label for all instances of B1 and B2 with a recall of 40% compared to 29% and 33% under BR and CC, respectively. The lower recall scores for the BR and CC models are due to the "unknown" labels, which were 16% and 7% of the total connections, respectively.

Moreover, LP predicts class labels with a hamming loss of 0.25, which is comparable to the loss of 0.28 and 0.26 achieved by BR and CC, respectively. However, the precision achieved by the LP model is 94.37%, less than the perfect 100% precision achieved by both BR and CC. This means that although some connections lack some labels with BR and CC, the predicted labels are 100% correct. The LP model succeeds in labeling all malware instances with comparatively more correct labels per instance (high recall) and with the least number of false labels in the prediction (low hamming loss). In summary, our models accurately identify zero-day traces with very low false positives and predict correct labels of the newer malware variants.

8 REVISITING ENTERPRISE LOGS: RESULTS

Due to the enterprise policy on data sharing, we were unable to obtain raw PCAPs to extract Tor cells for connection-level features. However, we can experiment using host-level features which can be extracted using Zeek logs alone.

We extracted and verified the Tor connections from the connection logs. We obtain a total of 207 Tor connections originating from nine unique source IPs. Out of the 207 connections, 197 originated from IP_A, three from IP_B, and the remaining seven connections originated from unique IPs each. To extract host-level features, we divided time into 6-minute epochs (since our trained model is based on PCAPs from 6-minute executions as well). This resulted in a total of 63 classification instances containing host-level features only from the connection logs. As listed in Table 2, we obtain host features from all categories except DNS-based features. This is because none of the 207 Tor connections had a corresponding log entry in the DNS logs. These were also excluded during classifier training for a fair evaluation of other host-level features when there are no onion domain leaks in DNS.

Results. Upon testing the classifier (E2 in Table 5) on extracted enterprise data features, 16 of the 63 instances were red-flagged as malicious with at least 80% confidence. All of these high confidence positive instances trace back to IP_A, which is also the same source with a total of 1,290 WannaCry-related C&C onion and kill switch domain lookups in the DNS logs. Setting the confidence threshold

to 70%, the classifier flags all the connections from host IP_A as malicious except two connections. Some C&C domain access attempts occur on the same day as the Tor connections (refer to Figure 6 in Appendix A for the timestamps of the Tor connections, onion domain leaks, and kill switch domain lookups). Finally, 29 instances were predicted malicious with significantly lower confidence of at most 64% originating from other source IPs with no evident DNS traces of malware activity.

9 RELATED WORK

Tor traffic analysis. To detect malicious Tor traffic, Ling *et al.* [42] propose TorWard, one of the earliest works which aim to detect botnet C&C, DoS, and spam at Tor exit nodes. This is done using Suricata IDS alerts. We focus on detecting Tor-based malware-generated traffic on the client side. Their detection approach relies on IDS alerts whereas we use traffic analysis on encrypted traffic. Fajana *et al.* [33] propose using circuit data visible to ORs to identify malicious traffic using simple traffic analysis features. However, this work lacks in providing sufficient details in data collection and experimentation methodologies, which fails to help understand the scale and realisticness of the experiments and prove the proposed claims. Cuzzocrea *et al.* [30] use traffic fingerprinting features such as packet timing and flow characteristics to classify Tor darknet traffic rather than malware traffic.

Some recent papers also focus on identifying Tor traffic and applications out of TLS flows. Sarkar *et al.* [57] use Deep Neural Network (DNN) on real-world Tor dataset achieving high accuracy in traffic classification and identifying its type such as web browsing, email, chat, etc. Similarly, Ma *et al.* [44] also use DNN on the same datasets though with a 2D CNN model for feature extraction. Iliadis *et al.* [39] and Chorood *et al.* [28] employ machine learning models to classify benign and darknet traffic. The former uses CICDarknet2020 public dataset and its provided features for classification, while the latter uses character frequencies in the Tor packet payload for classification. None of these works attempts to identify Tor-based malware traffic.

Website Fingerprinting attacks, known to de-anonymize user activity, are closely related to our work as we use a similar traffic analysis approach. The earliest work in WF was proposed by Sun *et al.* [63]. The authors studied the attack over TLS encrypted single-hop traffic. They measure the similarity between traffic signatures of webpages derived from the number and size of HTML objects using the Jaccard similarity metric. Herrmann *et al.* [38] developed the attack to work on single- and multi-hop Privacy Enhancing Technologies such as SSL/TLS, SSH, VPN, and Tor with Multinomial Naive Bayes using frequency distribution of IPv4 packet sizes. Several works followed thereafter over the years in attempts to enhance attack performance using advanced machine learning features, models, and realistic experimental datasets [26, 27, 37, 43, 53, 61, 62, 64, 65]. Each work advances the attack by improving the fingerprinting features (such as packet ordering, volume, time, and direction of packets) and the classification model. We use a superset of all these features included within the 150 proposed in the k-fingerprinting attack by Hayes *et al.* [37]. Although there exists a large body of work on malware detection in general [46, 49], we focus on comparing our work with network-based solutions.

Network-based malware detection. Some earlier works such as BotMiner by Gu *et al.* [35] and Jackstraws by Jacob *et al.* [40] detect C&C from botnet traffic. The former work applies a clustering-based approach to traffic similarity. The idea develops on the fact that bots within a botnet share similar C&C patterns. Jackstraws uses a similar approach as ours in utilizing machine learning for C&C traffic detection but uses behavior graphs (graph templates) based on end host system calls associated with TCP connections. Other works rely on communication order and patterns to identify malware families [50, 51]. More recently, Alahmadi *et al.* [22] propose BOTection, which uses machine learning over content agnostic flow-based feature vectors. They capture the frequency of change in connection states and burst connection behavior of flows generated by bots. The work uses Markov chains to model network behavior while we use features derived from fingerprinting Tor traffic. Piskozub *et al.* [55] group flows into flowsets with statistical features to detect suspicious flowsets and evaluate their approach on real-world traffic. Ghafir *et al.* [34] introduce BotDet, a system with detection modules that captures specific techniques used in botnet C&C communication. These works relate to ours in their goal of identifying malware C&C traffic using network traffic but differ from ours in terms of their approach. Moreover, these solutions operate on general network traffic while we specifically focus on Tor traffic.

10 CONCLUSION

More and more malware variants use Tor to hide their presence and evade detection. Due to the importance of Tor for benign usage, we explore the possibility of using traffic analysis Tor-based malware connections. We compile hundreds of thoroughly verified Tor-based malware binaries and deploy them for months in order to collect their traffic. We also collect benign traffic generated using the Tor browser in the same sandbox environment simulating different user profiles. Our classifiers are able to identify malware connections with high precision and recall and low FPR. Our models can identify the malware class based on its behavior. Finally, we show the effectiveness of our models in the face of fresh zero-day binaries, where we are able to identify all malware connections even when they constitute 5% of test traffic.

This work opens the door for various future directions. Improvements in malware sandboxing technologies can improve data and traffic collection, which in turn can address research questions such as can we identify each malware binary from its traffic? Can we identify if different binaries are connecting to the same C&C (thereby linking them to the same operators)? In our experience, various malware variants are quite elaborate in their evasion techniques [23, 24]. Detection based on traffic analysis can be thwarted by malware by adding benign traffic to their operation. In this case, adversarial machine learning defenses can address such nefarious tactics. Detection by ORs can also be effective against such adversarial defenses since ORs have access to clean circuit data.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive suggestions, comments, and valuable insights, which helped us improve

the paper. We also thank Saravanan Thirumuruganathan for the useful discussion and feedback.

REFERENCES

[1] 2013. How to Handle Millions of New Tor Clients. https://blog.torproject.org/how-handle-millions-new-tor-clients/#comment-34624.

[2] 2016. Tor-nonTor Dataset (ISCXTor2016). https://www.unb.ca/cic/datasets/tor.html.

[3] 2017. WannaCry Ransomware Campaign: Threat Details and Risk Management. https://www.fireeye.com/blog/products-and-services/2017/05/wannacry-ransomware-campaign.html.

[4] 2021. Hybrid Analysis. https://www.hybrid-analysis.com/.

[5] 2021. Tor Hidden Services Deprecation Timeline. https://blog.torproject.org/v2-deprecation-timeline/.

[6] 2021. Tor Metrics. https://metrics.torproject.org.

[7] 2022. Ahmia - Search Tor Hidden Services. https://ahmia.fi/.

[8] 2022. Autogluon Predictors. https://auto.gluon.ai/stable/api/autogluon.predictor.html?highlight=p_value.

[9] 2022. Autogluon Tabular Models (Documentation). https://auto.gluon.ai/stable/api/autogluon.tabular.models.html?highlight=weighted%20ensemble%20l2.

[10] 2022. AutoGluon Tasks. https://auto.gluon.ai/stable/api/autogluon.task.html.

[11] 2022. BinaryRelevance: scikit-multilearn. http://scikit.ml/api/skmultilearn.problem_transform.br.html.

[12] 2022. ClassifierChains: scikit-multilearn. https://scikit-learn.org/stable/auto_examples/multioutput/plot_classifier_chain_yeast.html.

[13] 2022. EternnalRocks-The Malware Wiki. https://malwiki.org/index.php?title=EternalRocks.

[14] 2022. Grayware- The Malware Wiki. https://malwiki.org/index.php?title=Grayware.

[15] 2022. LabelPowerset: scikit-multilearn. http://scikit.ml/api/skmultilearn.problem_transform.lp.html#skmultilearn.problem_transform.LabelPowerset.

[16] 2022. Python dpkt. https://dpkt.readthedocs.io/en/latest/.

[17] 2022. Spyware- The Malware Wiki. https://malwiki.org/index.php?title=Spyware.

[18] 2022. SystemBC – a RAT in the Pipeline. https://blogs.blackberry.com/en/2021/06/threat-thursday-systembc-a-rat-in-the-pipeline.

[19] 2022. Top 1M Alexa. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[20] 2022. Trojan- The Malware Wiki. https://malwiki.org/index.php?title=Adware.

[21] 2022. Zeek, An Open Source Network Security Monitoring Tool. https://zeek.org/.

[22] Bushra A Alahmadi, Enrico Mariconti, Riccardo Spolaor, Gianluca Stringhini, and Ivan Martinovic. 2020. BOTection: Bot Detection by Building Markov Chain Models of Bots Network Behavior. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. 652–664.

[23] Omar Alrawi, Moses Ike, Matthew Pruett, Ranjita Pai Kasturi, Srimanta Barua, Taleb Hirani, Brennan Hill, and Brendan Saltaformaggio. 2021. Forecasting Malware Capabilities From Cyber Attack Memory Images. In *30th USENIX Security Symposium (USENIX Security 21)*. 3523–3540.

[24] Omar Alrawi, Charles Lever, Kevin Valakuzhy, Kevin Snow, Fabian Monrose, Manos Antonakakis, et al. 2021. The Circle Of Life: A {Large-Scale} Study of The {IoT} Malware Lifecycle. In *30th USENIX Security Symposium (USENIX Security 21)*. 3505–3522.

[25] Athanasios Avgetidis, Omar Alrawi, Kevin Valakuzhy, Charles Lever, Paul Burbage, Angelos Keromytis, Fabian Monrose, and Manos Antonakakis. 2023. Beyond The Gates: An Empirical Analysis of HTTP-Managed Password Stealers and Operators. In *32nd USENIX Security Symposium (USENIX Security 23)*.

[26] Sanjit Bhat, David Lu, Albert Hyukjae Kwon, and Srinivas Devadas. 2019. Var-cnn: A Data-efficient Website Fingerprinting Attack based on Deep Learning. (2019).

[27] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 605–616.

[28] Pitpimon Choorod and George Weir. 2021. Tor Traffic Classification Based on Encrypted Payload Characteristics. In *2021 National Computing Colleges Conference (NCCC)*. IEEE, 1–6.

[29] Lucian Constantin. 2012. Tor Network Used to Command Skynet Botnet. https://www.computerworld.com/article/2493980/tor-network-used-to-command-skynet-botnet.html.

[30] Alfredo Cuzzocrea, Fabio Martinelli, Francesco Mercaldo, and Gianni Vercelli. 2017. Tor Traffic Analysis and Detection via Machine Learning Techniques. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 4474–4480.

[31] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium (USENIX Security 04)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router

[32] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv preprint arXiv:2003.06505* (2020).

[33] Oluwatobi Fajana, Gareth Owenson, and Mihaela Cocea. 2018. Torbot Stalker: Detecting Tor Botnets through Intelligent Circuit Data Analysis. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE, 1–8.

[34] Ibrahim Ghafir, Vaclav Prenosil, Mohammad Hammoudeh, Thar Baker, Sohail Jabbar, Shehzad Khalid, and Sardar Jaf. 2018. BotDet: A System for Real Time Botnet Command and Control Traffic Detection. *IEEE Access* 6 (2018), 38947–38958.

[35] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. 2008. Botminer: Clustering Analysis of Network Traffic for Protocol-and Structure-independent Botnet Detection. (2008).

[36] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 1187–1203.

[37] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 1187–1203.

[38] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. 2009. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In *Proceedings of the 2009 ACM workshop on Cloud computing security*. 31–42.

[39] Lazaros Alexios Iliadis and Theodoros Kaifas. 2021. Darknet Traffic Classification Using Machine Learning Techniques. In *2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. IEEE, 1–4.

[40] Gregoire Jacob, Ralf Hund, Christopher Kruegel, and Thorsten Holz. 2011. JACKSTRAWS: Picking Command and Control Connections from Bot Traffic.. In *USENIX Security Symposium*, Vol. 2011. San Francisco, CA, USA.

[41] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. 2017. Characterization of Tor Traffic using Time Based Features.. In *ICISSp*. 253–262.

[42] Zhen Ling, Junzhou Luo, Kui Wu, Wei Yu, and Xinwen Fu. 2015. Torward: Discovery, Blocking, and Traceback of Malicious Traffic over Tor. *IEEE Transactions on Information Forensics and Security* 10, 12 (2015), 2515–2530.

[43] Liming Lu, Ee-Chien Chang, and Mun Choon Chan. 2010. Website Fingerprinting and Identification using Ordered Feature Sequences. In *European Symposium on Research in Computer Security*. Springer, 199–214.

[44] Haoyu Ma, Jianqiu Cao, Bo Mi, Darong Huang, Yang Liu, and Zhenyuan Zhang. 2021. Dark Web Traffic Detection Method Based on Deep Learning. In *2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS)*. IEEE, 842–847.

[45] Brad Miller, Ling Huang, Anthony D. Joseph, and J. Doug Tygar. 2014. I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis. *ArXiv* abs/1403.0297 (2014).

[46] Abedelaziz Mohaisen and Omar Alrawi. 2013. Unveiling Zeus: Automated Classification of Malware Samples. In *Proceedings of the 22nd International Conference on World Wide Web*. 829–832.

[47] Aziz Mohaisen and Omar Alrawi. 2014. Av-meter: An Evaluation of Antivirus Scans and Labels. In *International conference on detection of intrusions and malware, and vulnerability assessment*. Springer, 112–131.

[48] Aziz Mohaisen, Omar Alrawi, Matt Larson, and Danny McPherson. 2013. Towards a Methodical Evaluation of Antivirus Scans and Labels. In *International Workshop on Information Security Applications*. Springer, 231–241.

[49] Aziz Mohaisen, Omar Alrawi, and Manar Mohaisen. 2015. AMAL: High-fidelity, Behavior-based Automated Malware Analysis and Classification. *computers & security* 52 (2015), 251–266.

[50] Aziz Mohaisen, Omar Alrawi, Andrew G West, and Allison Mankin. 2013. Babble: Identifying Malware by its Dialects. In *2013 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 407–408.

[51] Aziz Mohaisen, Andrew G West, Allison Mankin, and Omar Alrawi. 2014. Chatter: Classifying Malware Families using System Event Ordering. In *2014 IEEE Conference on Communications and Network Security*. IEEE, 283–291.

[52] Palo Alto Networks. 2012. Threat Assessment: Egregor Ransomware. https://unit42.paloaltonetworks.com/egregor-ransomware-courses-of-action/.

[53] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website Fingerprinting in Onion Routing based Anonymization Networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. 103–114.

[54] Eva Papadogiannaki and Sotiris Ioannidis. 2021. A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures. 54, 6 (2021). https://doi.org/10.1145/3457904

[55] Michal Piskozub, Riccardo Spolaor, and Ivan Martinovic. 2019. Malalert: Detecting Malware in Large-scale Network Traffic using Statistical Features. *ACM SIGMETRICS Performance Evaluation Review* 46, 3 (2019), 151–154.

[56] Ferry Astika Saputra, Isbat Uzzin Nadhori, and Balighani Fathul Barry. 2016. Detecting and Blocking Onion Router Traffic Using Deep Packet Inspection. In

2016 International Electronics Symposium (IES). IEEE, 283–288.

[57] Debmalya Sarkar, P Vinod, and Suleiman Y Yerima. 2020. Detection of Tor traffic using Deep Learning. In *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*. IEEE, 1–8.

[58] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2017. Beauty and the Burst: Remote Identification of Encrypted Video Streams. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1357–1374. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/schuster

[59] Silvia Sebastián and Juan Caballero. 2020. AVclass2: Massive Malware Tag Extraction from AV Labels. In *Annual Computer Security Applications Conference* (Austin, USA) *(ACSAC '20)*. Association for Computing Machinery, New York, NY, USA, 42–53. https://doi.org/10.1145/3427228.3427261

[60] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. 2016. AVclass: A Tool for Massive Malware Labeling, Vol. 9854. 230–253. https://doi.org/10.1007/978-3-319-45719-2_11

[61] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1928–1943.

[62] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. 2019. Triplet Fingerprinting: More Practical and Portable Website Fingerprinting with n-shot Learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1131–1148.

[63] Qixiang Sun, Daniel R Simon, Yi-Min Wang, Wilf Russell, Venkata N Padmanabhan, and Lili Qiu. 2002. Statistical Identification of Encrypted Web Browsing Traffic. In *Proceedings 2002 IEEE Symposium on Security and Privacy*. IEEE, 19–30.

[64] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 143–157.

[65] Tao Wang and Ian Goldberg. 2013. Improved Website Fingerprinting on Tor. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. 201–212.

[66] Tao Wang and Ian Goldberg. 2013. Improved Website Fingerprinting on Tor. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. 201–212.

[67] Tao Wang and Ian Goldberg. 2016. On Realistically Attacking Tor with Website Fingerprinting. *Proc. Priv. Enhancing Technol.* 2016, 4 (2016), 21–36.

[68] Charles V. Wright, Lucas Ballard, Fabian Monrose, and Gerald M. Masson. 2007. Language Identification of Encrypted VoIP Traffic: Alejandra y Roberto or Alice and Bob?. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium* (Boston, MA) *(SS'07)*. USENIX Association, USA, Article 4, 12 pages.

[69] Yixiao Xu, Tao Wang, Qi Li, Qingyuan Gong, Yang Chen, and Yong Jiang. 2018. A Multi-tab Website Fingerprinting Attack. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 327–341.

A WANNACRY LEAKS IN REAL WORLD ENTERPRISE DATA

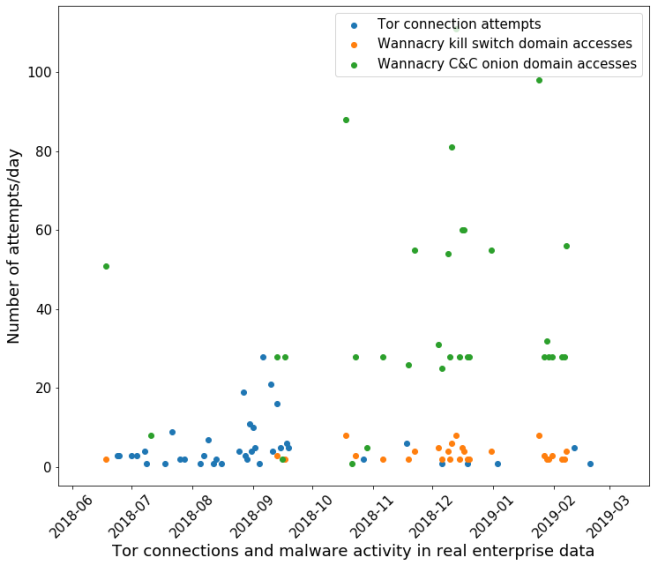

Figure 6: Timeline of Tor connections and WannnaCry (ransomware) C&C and kill switch onion domain accesses in real enterprise network

B BINARY-LEVEL SPLITTING

In Section 5.2, we explained a training approach where we split at the PCAP level. For example, in D10, each binary will have seven PCAPs in training and three in testing. This provides the classifier with better training and visibility as it allows it to train on more unique binaries. We have also experimented with a more conservative approach where we split by binaries first. In this way, PCAPs belonging to a certain binary are not split between training and testing. In D10, all 10 PCAPs associated with a binary are used for training or testing. Connections are then extracted and features computed in a similar manner to what is described in Section 5.2. Table 8 summarizes our results for the binary-level splitting approach using our experimental malware datasets. FPR is clearly increasing with the decreasing number of unique binaries for each dataset even though training instances per binary are increasing. However, for D5, FPR is comparable to PCAP-level split, though with a slightly lower recall. The other datasets have higher recall but also higher FPR. D5 appears to provide the best training to the classifier as it can identify the majority of malware instances with the lowest FPR.

Table 8: Performance comparison of different datasets using binary splitting for model training/testing

Dataset	Precision(%)	Recall(%)	FPR(%)	AUC(%)
D5	94.0	77	0.78	98.41
D10	95.15	83.16	1.15	98.74
D20	96.39	85.70	1.43	96.27
D30	91.20	88.8	3.2	98.02

C EXCLUDED MALWARE FAMILIES

Table 9: Malware families and their binaries which did not produce Tor traffic in the sandbox

Malware family	No. of binaries
mokes	19
hype	6
ficker	2
crysan	5
zenpack	4
smokeloader	7
safebits	1
neshta	3
vkontaktedj	3
hesv	1
pioneer	2
ursnif	1
predator	1
nsisinject	1
tofsee	1
zard	1
fugrafa	4
rugmi	1
karo	1
zerodloader	1
daws	1
torjok	1
penguish	1