Detection of Anonymised Traffic: Tor as Case Study

Bruno Dantas[1], Paulo Carvalho[1(✉)], Solange Rito Lima[1], and João Marco C. Silva[2]

[1] Centro Algoritmi, Universidade do Minho, 4710 057 Braga, Portugal
a74207@alunos.uminho.pt, {pmc,solange}@di.uminho.pt
[2] HASLab, INESC TEC, Universidade do Minho, Braga, Portugal
joao.marco@inesctec.pt

Abstract. This work studies Tor, an anonymous overlay network used to browse the Internet. Apart from its main purpose, this open-source project has gained popularity mainly because it does not hide its implementation. In this way, researchers and security experts can fully examine and confirm its security requirements. Its ease of use has attracted all kinds of people, including ordinary citizens who want to avoid being profiled for targeted advertisements or circumvent censorship, corporations who do not want to reveal information to their competitors, and government intelligence agencies who need to do operations on the Internet without being noticed. In opposition, an anonymous system like this represents a good testbed for attackers, because their actions are naturally untraceable. In this work, the characteristics of Tor traffic are studied in detail in order to devise an inspection methodology able to improve Tor detection. In particular, this methodology considers as new inputs the observer position in the network, the portion of traffic it can monitor, and particularities of the Tor browser for helping in the detection process. In addition, a set of Snort rules were developed as a proof-of-concept for the proposed Tor detection approach.

1 Introduction

Privacy is a human right and online privacy should be no different. While communications and data need firm online protections, bureaucracy has been slow to respond to the pace of technological changes. The lack of trust in the Information Technology (IT) domain has led individuals to discover different ways of hiding their online identities (online anonymity). The main argument against online anonymity by governments is about users having a lack of accountability. In other words, anonymity can harbour criminal activity by making the tracing of online activities more difficult [1]. Further, anonymous traffic hardens the management and monitoring of network infrastructures because the traffic cannot be easily associated with its original sources and/or destinations. Nevertheless, detecting and blocking anonymous traffic may, in some cases, be crucial for the good operation of the network.

O. Galinina et al. (Eds.): NEW2AN 2020/ruSMART 2020, LNCS 12526, pp. 95–109, 2020.
https://doi.org/10.1007/978-3-030-65729-1_9

This work studies Tor, a tool that allows its users to achieve online anonymity. By using Tor, users can access the public Internet without worrying about censors, governments, service providers, and so on. The motivation of this work is to take the area of anonymous traffic detection one step further in its continuous research. In particular, this paper starts by dissecting the main Tor architectural components and operation in order to establish an encompassing methodology for its detection. This involves a systematic coverage of relevant aspects to consider in the traffic inspection strategy, such as the observer's position in the network, the portion of traffic under analysis and Tor traffic characteristics at distinct network layers. As proof-of-concept, after exploring and identifying Tor behaviour and particularities, a set of Snort rules was created to detect Tor traffic.

This paper is organised as follows: a discussion on traffic characterisation, anonymity systems and other related works is carried out in Sect. 2; in Sect. 3, the methodology for detecting Tor traffic is discussed from different perspectives and protocolar levels. As proof-of-concept, the definition of Snort rules for Tor detection and the experimental results are provided and discussed in Sect. 4. Finally, the main conclusions of the study are included in Sect. 5.

2 State-of-the-Art

2.1 Traffic Classification

The discipline of traffic classification tries to associate traffic flows or packets with the applications, or application types, that generated them. In the context of computer networks, the five tuple (destination/source Internet Protocol (IP) address, destination/source port number, protocol field) is commonly used to (uniquely) identify different flows, and is used since the first-generation firewalls [2].

While in the early days of the Internet port-based approaches (checking packets' port numbers) was enough to achieve high accuracy classification results [3], over the last two decades, some developments make it difficult for operators and service providers to classify traffic flows: applications that have no IANA registered ports; the use of well-known ports to circumvent filtering; physical servers may offer services through the same public address but on different ports.

Operators were then forced to use another approach, commonly called Deep Packet Inspection (DPI), by looking at packets' content to discover the application being used [4]. This approach has two downsides. The use of pattern matching can become easily slow because each incoming packet has to be compared with thousands of different signatures. Also, end-to-end encryption is becoming ubiquitous, which makes DPI less effective [5]. Nevertheless, relevant information can be extracted from encrypted connections, typically from the session's initiation. The reason for this is that security protocols usually have an initialisation phase that is not encrypted.

With the increasing complexity of networks, the classification methods are usually supported with the help of protocols knowledge [5], coupling traffic classification with other approaches, such as machine learning, deep learning, and

specific heuristics [3,5]. These approaches are mainly applied to specific network metrics depending on each application characteristics, at different granularity levels, being also referred as host behaviour and statistical classification. Despite this, if operators' techniques are publicly known, developers intentionally change the application's network metrics (e.g. packets' inter-arrival times, packets' length, using dynamic ports). By modifying them, it is possible to obfuscate the application generating the traffic.

2.2 Online Anonymity Systems

The main goal of anonymity systems is to avoid traffic analysis and network surveillance, and to block any tracking of users' identities in the Internet [6]. Why is it needed, if almost all applications' data is encrypted? First, service providers usually have logging systems to monitor their infrastructure. The recorded logs can have information that can be used to keep track of users' identities or activities. Secondly, secure communication protocols can reveal information or have implementation flaws. For example, the Client Hello message in the Transport Layer Security (TLS) handshake can carry the Server Name Indication in plain text (SNI extension). Thirdly, as already mentioned, traffic classification techniques are currently adopting statistical approaches to classify encrypted traffic, based on network and transport layers information, communication metrics and user behaviour. In other words, it is still possible to guess, with some level of certainty, what kind of services users are accessing, even if the communications are encrypted. These discussed points are the reasons why most of the anonymity systems use techniques in network or transport layers. The goal is to separate the application data from the lower level layers as a way to obfuscate who is accessing the service. Typically, this is achieved by bouncing the traffic to an intermediate entity before accessing the final service. That entity then uses some strategy to obfuscate the traffic (discussed next).

Depending on the architecture, anonymity systems can be used to circumvent geographically blocked content, dodge targeted marketing or test network attacks. Also, they are a troublesome for governments in censorship countries, as they allow citizens to access censored websites without being discovered. As a consequence, those countries' government agencies and particular service providers block traffic generated or sent to those tools. When an anonymity system starts to become more famous, the simplest approach is to block their public IP addresses [7]. In this way, citizens are forced to access the service without an anonymity system or to find other system less famous and not blocked yet. Without directly blocking anonymity systems, these agencies would have to apply other traffic classification techniques already discussed. If the traffic is classified as belonging to some anonymity system and is blocked, citizens are unable to access whatever they like through that system (even if, the content is legal).

Currently, there is no anonymity system capable of leaving all parties satisfied. These systems use techniques such as proxies, mix networks, tunnelling and overlay networks. Each one has its own design, which means that each one has its specific use case, advantages and disadvantages. This paper will focus on

Tor, an anonymity system that uses an overlay network to provide anonymous browsing to its users.

2.3 Tor Project

Tor (or Tor Project) is a continuously growing open-source project that provides anonymity at network/transport layers over the TCP/IP protocol stack. Its first release was in 2002 with the name The Onion Router (hence its actual abbreviated name). More specifically, Tor is a circuit-based low-latency anonymous communication service that provides perfect forward secrecy, congestion control, directory authorities, integrity checking, configurable exit policies, and a practical design for location-hidden services via rendezvous points [8].

The service is circuit-based because before sending any data, it creates a route of nodes through which the data will pass. Depending on the position in the circuit, each node can be called entry/guard, middle or exit node. Instead of nodes, these can also be called onion routers or relays. Tor provides a low-latency service because it was deployed to be used in interactive or non-linear environments, such as web browsing. Perfect forward secrecy means that once the session keys are deleted, subsequently compromised nodes cannot decrypt old traffic (unique session keys are created for different sessions). Tor uses an incremental path-building design, where the initiator negotiates session keys with each successive hop in the circuit, used for encrypting the message. Each onion router in the path then strips off a layer of encryption, until the exit node decrypts the last layer and becomes able to redirect the original message's request. Figure 1 illustrates this process with only two onion routers (in real operation there are at least three).

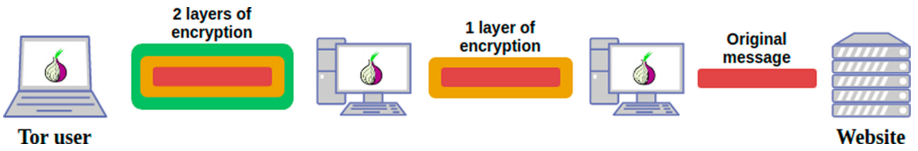

Fig. 1. Example of Tor encryption layers.

Congestion control is also implemented to prevent onion routers from getting congested. Directory authorities are trustworthy servers responsible for retrieving control information to clients (e.g., list of onion routers that will make up the circuit). Integrity checking is performed in two stages. First, onion routers use the TLS protocol to communicate with each other (see details in Sect. 3.1). Second, specific Tor messages called relay cells contain an end-to-end checksum for integrity checking. Configurable exit policies, as the name suggests, is a feature that enables users to circumvent problems with service usage (e.g., selecting the country of the last onion router in the circuit). Location-hidden services provide anonymous services to other users (users remain anonymous too), such as e-commerce, news or illegal activities [9].

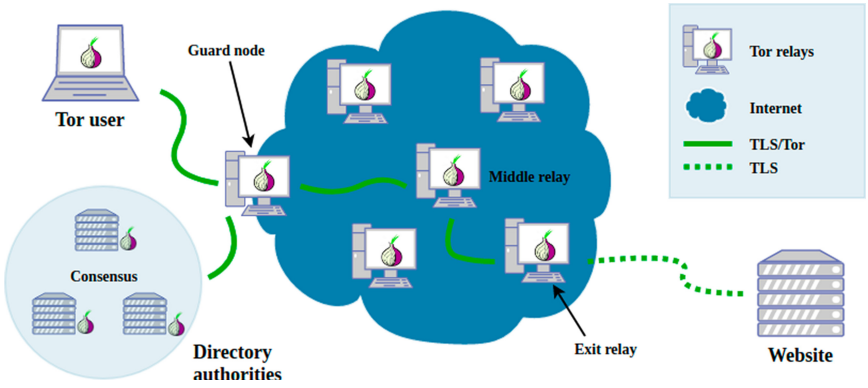

Fig. 2. Tor architecture review and related terminology.

A typical architecture of Tor is depicted in Fig. 2. At its core, Tor is simply a tool that can build paths given a set of routers [7]. One relay is chosen more frequently, the guard node, which is highlighted in the figure. On the one hand, this relay is a special trustworthy node that always acts as the entry relay, as long as it is not compromised or (periodically) rotated. On the other hand, middle relays and exit relays are chosen proportionally to their available bandwidth[1]. There are fewer exit relays than middle relays because Tor allows volunteers to run either as middle or exit relays. The reason is that exit relays do not know what websites are being accessed by users, and so they are more vulnerable to attacks if the websites are (intentionally) compromised [8].

The directory authorities are responsible for distributing control information across the (overlay) network. Basically, they agree on a consensus, a compressed document, so that each relay can check its validity. Communications within Tor use their own defined encryption techniques(over a TLS layer).

2.4 Related Work

In [11], machine learning techniques are used for traffic classification in three different anonymity tools: Tor, JonDonym and I2P. Using the *Anon17* dataset[2], the authors measure three different levels of granularity to distinguish between the anonymity tool, traffic type and application. The results showed high accuracy, progressively reduced for each granularity level, evaluated after receiving the flows' eighth packet (as the beginning of a flow contains the negotiation of parameters and requests between client/server or peers, with protocol-specific headers and message sequences [3]). Despite this, the authors show the maximum

[1] The bandwidth is constantly tested by directory authorities to prevent attacks where relays claim to have more bandwidth than they really have (to be more frequently chosen) [8,10].

[2] https://web.cs.dal.ca/~shahbar/data.html.

achieved accuracy and F-measure, along with the number of packets needed to sustain those results. The effects of feature importance (up to 74) and temporal-related features to the network are also investigated.

The work reported in [12] demonstrates the presence of *hot exit points* in Tor. These are exit-nodes agglomerations, controlled by some Internet Service Providers (ISPs), that are almost always used, despite the existence of other exit-nodes options. The results are based on 1.5 years of recorded data, being a threat to anonymity (as ISP can correlate traffic more easily).

A different Tor design is proposed in [13]. By introducing group signatures, a cryptographic technique, it is possible to distinguish between legitimate and illegitimate users. When some malicious action is performed by a user, it is possible to block or denounce him. This approach is not perfect. Although relay volunteers and ISPs increase trust in the overlay network, benign users have to trust the entity in charge of blocking or denouncing them (basically the same problem of mixing proxies).

The identification of TLS abnormalities in Tor is provided in [14]. The work begins by identifying TLS characteristics in Tor (e.g., certificate extensions). Then, Snort rules are created according to those characteristics, which could effectively identify the presence of Tor traffic. However, at the time of writing, Tor only supported TLS 1.0, resulting in outdated rules when considering the latest TLS 1.3 and 1.2 versions.

The work in [15] begins by identifying the most known attacks that could be used to deanonymise Tor circuits. The two outlined categories of attacks are traffic correlation and webpage fingerprinting. The vulnerable identified areas are the guard-node selection and rotation algorithm, the intercell transmission timings and other traffic metrics (cell order, amount, interval, size and direction). Then, the authors have focused on these areas to collect the proposed and already-implemented countermeasures that could enhance Tor resistance as regards the three mentioned vulnerabilities. Finally, the process of evaluation is accomplished by comparing each countermeasure with a set of (security) requirements previously defined (following the *MoSCoW* method).

A longitudinal study of Tor network is presented in [16]. The work is based on a passive analysis of TLS traffic over more than three years in four large universities. The results show that it is possible to identify Tor, specifically through some information present in X.509 certificates and other exchanged parameters within the TLS handshake. Despite this, it is assumed that Tor's detection will remain an arms race.

Although the mentioned studies address the problematic of Tor traffic classification and/or detection of malicious behaviours at some extend, important aspects of an encompassing methodology for Tor inspection and subsequent detection are not addressed as a whole. This motivates the present study and the methodological discussion presented below.

3 Inspecting Tor Traffic

This section is dedicated to discuss important Tor inspection methodology aspects namely, the observer's position in the network, the portion of traffic it can monitor, protocol layer issues and related metrics, and particularities of Tor browser.

3.1 Observer Position

As regards the observer position in the network, a Tor detector might be placed at three distinct points, i.e., between: (i) an onion proxy and a guard node; (ii) two onion routers (or a guard node and an onion router); and (iii) an exit node and the final destination.

(i) Onion proxy and guard node - an external observer monitoring any link between the onion proxy (or a Tor browser) and the guard node can infer relevant information. First, the (partial) location can be exposed through the inspection of the guard node's IP addresses involved in the connection. Second, it is possible to identify whether the client is using Tor. This detection is performed by inspecting the IP address and comparing it to known guard nodes[3], or inspecting the TLS handshake (assuming that the connection was not already established).

(ii) Two onion relays - this case and the previous one follow different traffic patterns. Although the TLS handshakes are slightly different, an external observer cannot distinguish them. In fact, one might think that it is easy to distinguish the two handshakes as in case (i) only one party authenticates himself (the guard node) and, in this case, both parties need to authenticate (to prevent impersonation attacks). As shown in Fig. 3, Tor handles this by performing a TLS handshake first, always as a server-only authentication handshake, and then performs an inner handshake to complete the authentication. To better illustrate this, consider two onion relays willing to communicate. At start, a normal client/server TLS handshake is performed, and only one of them (the next in the circuit) will authenticate itself using a local certificate. Only after this, inside the TLS session just created, the authentication will be completed in an inner handshake, specified by Tor (not a TLS handshake). In this inner handshake, the parties will agree in a common version, exchange certificates, and other parameters. To an external observer, the outer handshake is the same for those two cases.

After the handshake phase, traffic patterns can be slightly different. Only the first case (onion proxy and guard node) uses connection-level padding. Any other connection does not use it (a connection to a bridge is treated as the first case[4]). There are proposals for changing the circuit-level padding between onion

[3] Tor Bridges. In that case, the onion proxy connects to a bridge before the guard node and can obfuscate itself from his regional ISP, i.e., the regional provider can only conclude that the onion proxy was communicating with another peer.

[4] https://gitweb.torproject.org/torspec.git/tree/padding-spec.txt.

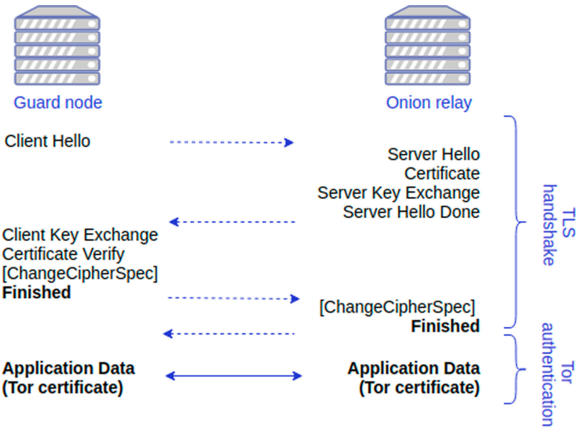

Fig. 3. Tor relays authentication

relays, however, they are not yet implemented. In connection-level padding, cells carry the same value as always, but new dummy cells are introduced to obfuscate traffic from external observers. Therefore, in theory, case (ii) leads to different traffic patterns from case (i).

(iii) Exit node and destination - from the exit node to the final destination, despite being a traditional HTTP/HTTPS connection, traffic can be analysed to possibly detect Tor presence. One way is to inspect the IP addresses involved in the connection and compare them to a list of known Tor (exit) nodes. Other possible way is to measure network metrics and infer that it is a Tor exit node if the delay of packets is higher than the normal. In this case, the packets' length is not fixed. In fact, the size of messages follows the normal operation of the HTTP protocol, but the exit node readjusts them to fit within the fixed required size (512 bytes) in the Tor network.

This knowledge might be helpful, for example, if a website administrator does not want Tor users to access his website. One possible reason for an administrator to want to block Tor under these conditions is that the information collected is no longer related to the original client's IP address, being now related to the exit node. Worse than that, a different client may use the same exit node, but they look the same to the website as both requests stem from the same IP address.

3.2 Observer Traffic Portion

In normal conditions, a passive observer can only monitor one link. However, monitoring more than one (overlay) link in Tor can be only achieved if that observer spans different countries. In fact, Tor assumes that does not have a defence against a global adversary [8]. By global adversary is meant a passive observer that can monitor more than one link, thus correlating flows and inferring the circuit that an onion proxy is using. Of course, one global adversary can

be made of smaller adversaries working together. Either in real-time or from recorded logs, it is possible to aggregate that data and correlate it to know what website a client was accessing through Tor.

3.3 Traffic Characteristics

An external observer may inspect packets payload, however their encryption do not allow retrieving useful information (at least individually). Because of that, inspection can only occur at lower protocolar levels, such as network, transport and session layers. In addition, measuring network parameters can help detecting Tor traffic, where the most important metrics are the length and the delay of packets. The former is useful because Tor uses fixed-size messages of 512 bytes (connection establishment cells are variable-sized). The latter can suggest the presence of Tor because delay values will typically be higher than normal (packets have to traverse onion relays across different countries). If possible, these metrics should be used in conjunction with other indicators or rules for completeness (e.g., network congestion may produce a false positive).

At network layer, IP is the first protocol worth to be inspected. It is known that some ISPs are already blocking static IP addresses used by directory authorities, onion relays or guard nodes (fact that led to the introduction of Tor Bridges). In addition, the websites providing information regarding current active onion relays, their associated IP addresses, geographical location, and sometimes the volunteer's personal information (Twitter profile) are plentiful. Even with the introduction of bridges, it is assumed that there is no magic bullet for their discovery. Although it is more difficult to enumerate all the bridges IP addresses, their discovery is indeed possible [17].

At transport layer, the Tor Browser currently uses random ports to communicate with the guard node, while onion proxies use the port determined by guard nodes. The same applies to fetch network information through other relays (up to three connections). Each relay can optionally act as a directory authority to help in the distribution of the consensus document, but the ports are randomly chosen as well. Note that the randomness of ports at transport layer is useless from a signature-based Intrusion Detection System (IDS) point of view. However, an anomaly-based IDS could probably deduce information if some ports were used more than others.

At session layer, in the case of TLS, the followed approach is simple. The objective is trying to detect fields and parameters that are different from other TLS flows (web browsers). By comparing the Tor browser with other browsers, one can start ruling out other TLS sources until only the Tor browser is left. According to [16], there are several fields within TLS that can be used to detect Tor. The present work takes that list as a starting point to discuss which TLS fields are useful for creating Tor detection rules. In this context, the TLS messages `Client Hello` and `Server Hello` were identified as relevant elements for sensing Tor.

3.4 Tor Browser Particularities

As mentioned, Tor traffic characteristics can be derived from the Tor browser TLS handshaking phase.

Starting with the `Client Hello` message, the first parameter to inspect is the TLS version. As most browsers already support TLS 1.3, every website connection within the browser will use an extensive list of TLS extensions in `Client Hello` messages, when compared with the previous version. Although this extensions list may be useful, inspecting the TLS `version` field is useless as both versions (v1.2 and v1.3) use the value `0x0303` for backward compatibility. As regards the `Session ID` field, it can be helpful as Tor nodes do not resume sessions, conversely to common services. Regarding cipher suites information, a major differentiating parameter is the `Cipher Suites Length`. Despite this, if this value is the same for Tor and other browsers, a comparison can be established based on `Cipher Suites List` contents. Table 1 compares two of the discussed fields in the `Client Hello` message for Tor and the most used browsers according to *w3schools* [5].

Table 1. Client-side TLS parameters

Browser	Cipher suites	Extensions
Tor	14	6
Chrome	17	17
Safari	23	8
Edge	19	10
Firefox	18	14
Opera	17	17

The `Server Hello` handshake message also carries useful information, however, its scope is broader than in the client. Therefore, this message should only be inspected to complement the `Client Hello`, as the number of different servers is very large, and each server has to follow the client behaviour. Note that considering `Server Hello` messages implies a stateful detection system [2] due to a potentially high number of sessions being handled concurrently (which may affect the trade-off between accuracy and performance). The `extensions` field can be a major field in the detection process, as Tor nodes use shorter extension lists than common servers (usually with a length of 13 or 18).

4 Experimental Evaluation

Based on the specific characteristics of Tor traffic presented above, Snort rules were defined and evaluated under distinct test scenarios. These rules allow to

[5] https://www.w3schools.com/browsers/default.asp.

introduce simple and effective detection in existing network systems, e.g., IDS or firewalls. The configuration details and the obtained results are discussed in the following sections.

4.1 Snort Rules

To explore the detection of Tor traffic, Snort rules were created targeting specific characteristics of TLS messages as explained in Sect. 3.4. To do so, a first rule (presented as Listing 1) triggers for every TLS packet sent from the internal network `$HOME_NET` to any external network having the `Cipher Suites Length` field with the value 28 (14 cipher suites supported).

```
alert tcp $HOME_NET any -> any any \
    (content:"|16|"; offset:0; depth:1; \
     content: "|00 1c|"; offset:44; depth:47; \
flowbits:set,tor_browser; flowbits:noalert; sid:1000001)
```

Listing 1: Snort rule: setting initial state

Note that an alert will not be generated for each packet matching this rule. Instead, the `tor_browser` variable will be set with the `flowbits:set` rule option. The alert is only generated if the response packet matches one of the two rules following this one (which is the purpose of the rule option `flowbits:isset`). These two rules are illustrated in Listing 2.

```
alert tcp any any -> $HOME_NET any \
        (msg: "TOR BROWSER DETECTED ! ! !"; \
        content:"|16|"; offset:0; depth:1; \
        content:"|00 0d|"; offset:47; depth:50; \
        flowbits:isset,tor_browser; sid:1000002)
alert tcp any any -> $HOME_NET any \
        (msg: "TOR BROWSER DETECTED ! ! !"; \
        content:"|16|"; offset:0; depth:1; \
        content:"|00 12|"; offset:47; depth:50; \
        flowbits:isset,tor_browser; sid:1000003)
```

Listing 2: Snort rules: detecting Tor

Basically, these rules will match response (`Server Hello`) packets which have the `Extension Length` field with the values 13 or 18. Within the rules, these values are represented in hexadecimal. Note that, the assumption that only `Client Hello` or `Server Hello` messages can match the rules can be violated as the protocol over TCP can be other than TLS. However, it is unlikely that a different protocol exactly matches the `content` rule options specified.

For the specific case of Snort, these rules might be enhanced by resorting to the SSL preprocessor, which allows the IDS to recognise a TLS session using the built-in `ssl_state` rule which matches every packet with `ssl_state: client_hello` or `ssl_state: server_hello` values.

4.2 Test Scenarios

In order to assess the proposed rules, background traffic was generated along with Tor connections following two test approaches: a simpler one, based on control traffic generated from non-anonymous browsers; and a more realistic test, in which Tor connections were introduced in real traffic captured in a campus network.

Tor connections were generated resorting to a script executing multiple instances of the Tor Browser Bundle, and connecting them to the Tor network, each time to a different guard node. To prevent from always connecting with the same guard node, each Tor Browser instance was set with a different list of wanted `EntryNodes`. Such node variability provides a more relevant test scenario, as it can detect guard nodes using different handshake protocols. Both background traffic and Tor specific traffic were merged using `tcpdump` and then processed by Snort, which applied both the community and the proposed rules. Mixing the general rules with Tor detection specific rules allow to address eventual collision with intrusion detection processes in operational networks. Details regarding the two test scenarios are:

(i) Browsers Traffic: In this first scenario, through a Python script, multiple connections to the fifty more accessed websites in Portugal[6] were established, using five commercial non-anonymous browsers (mentioned in Table 1) in a local network. The purpose of using different browsers is to have `Client Hello` messages varying the cryptographic parameters. In this way, at least 250 handshakes are performed (some websites perform more than one).

(ii) Real Traffic: In this scenario, a `Sophos XG 105 Firewall` was deployed for sniffing all traffic generated by multiple users sharing a LAN during one hour. The traffic passing through the firewall does only include each user's laptop in that LAN. The corresponding network traffic trace includes various applications, such as *OpenVPN*, *CiscoVPN*, *Microsoft OneDrive*, *Microsoft Teams*, *Skype*, *Slack*, *Spotify*, *EMC Avamar*, *Tortoise SVN*, *NetBeans* or *Maven*, resulting from a normal user activity.

Although performed in an offline environment (to prevent connection variations or instabilities across multiples sessions), all tests assume a monitoring point placed between the onion proxy and guard node (see Sect. 3.1).

4.3 Evaluation Results

The accuracy of the rules created is measured evaluating whether the number of Snort alerts triggered by Tor traffic is equal to the number of connections established through the script. False positives and false negatives are identified by comparing all processed packets with the trace containing only the Tor traffic.

For test scenario (i), with traffic generated by five different non-anonymous browsers, the detection rules led to 100% accuracy. In fact, some alerts were

[6] https://www.similarweb.com/top-websites/portugal.

triggered only by executing the Tor browser, which demonstrates their effectiveness in identifying Tor activity even before the user request any resource. This is possible as the Directory authority activity (see Sect. 2.3) follows the same TLS pattern as the relay nodes.

Considering the second and more realistic scenario, the proposed rules were also able to identify all connections originated from a Tor browser. However, in this case, it was also identified a significant number of false positives, 25%. Although being high, all the false positives were related to the same traffic type, namely *Microsoft Teams* desktop application.

A deeper inspection of false positives revealed they were caused by the way Snort processes packets. Basically, the proposed rules have a specific offset because when analysing Tor traffic, no session identifiers were present. Without session identifier, the `Client Hello` message has only one field in this context, the `Session ID Length` with the value zero. When this identifier is present, the `Client Hello` message also includes the length field with the corresponding value. Hence, when creating the rules, the field following the `Session ID Length` was not its actual value, but the next field, i.e., the `Cipher Suites Length`. All connections erroneously classified as Tor traffic had a session identifier which was causing a no alert triggering. However, the alerts were generated, so it is assumed that Snort can ignore the identifier value or shift the offset accordingly. A strategy that might prevent this type of false positive is the introduction of additional parameters into the rules. For instance, the SNI extension in the `Client Hello` might be analysed and compared either to a list of trusted server names or to a list of previously enumerated services.

Figure 4 (left) illustrates the total number of packets of test scenarios (i) and (ii), and the number of packets that Snort analysed. Although the analysis was not performed in real-time, processing the real traffic scenario efficiently required significant resources (it was analysed in less than 3 s, which corresponds to approximately 8 Gbps). This ratifies the importance of establishing simple detection rules, as adding new fields to be matched and/or more chained rules applied to all packets might be computationally prohibitive. In Fig. 4 (right), the number of Tor alerts and logs are expressed. As shown, Snort could correctly log all the generated alerts, however, the network trace with real traffic lead to 25% of false positives in detecting Tor traffic.

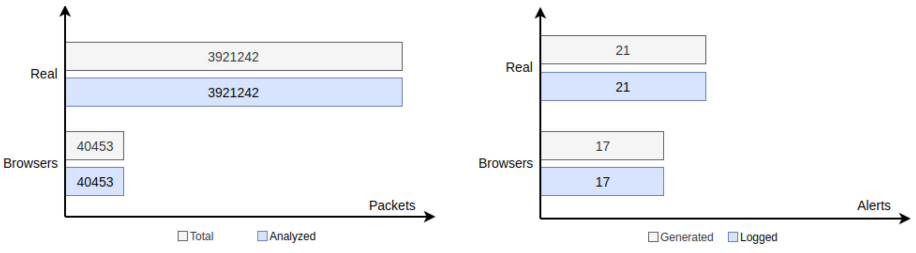

Fig. 4. Number of packets per test (left); Number of Tor alerts per test (right)

5 Conclusions

Within anonymisation systems, Tor has gained popularity as an anonymous overlay network used to browse the Internet. The present work has provided a comprehensive study exploring particular characteristics of Tor traffic in order to allow a simple and effective detection in production networks. The proposed Tor inspecting methodology considers new aspects to be considered for helping in the detection process, as the observer position in the network, the portion of traffic it can monitor, and the session-level particularities of the Tor browser. The proof-of-concept has involved the definition of a set of Snort rules tested in distinct traffic scenarios for Tor detection. In both scenarios, the results show the effectiveness of simple rules in detecting Tor traffic with high accuracy.

With respect to this area of research, it will remain a strong competition among the entities in search for online anonymity and the ones that search for surveillance and/or censorship. Currently, there is no one-fits-all solution to this problem. Different strategies have arisen due to the increasing interest in this area, but those only solve the problem to some extent. Most importantly, the anonymous systems research line must continue to evolve to allow users online privacy while allowing means of service providers and system administrators being able to control their infrastructures. This study represents a step forward in this direction.

Acknowledgements. This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project UIDB /50014/2020, and by FCT – Fundação para a Ciência e Tecnologia – within the R&D Units Project Scope: UIDB/00319/2020.

References

1. Winkler, S., Zeadally, S.: An analysis of tools for online anonymity. Int. J. Pervasive Comput. Commun. **11**(4), 436–453 (2015)
2. Neupane, K., Haddad, R., Chen, L.: Next generation firewall for network security: a survey. In: Conference Proceedings - IEEE SOUTHEASTCON 2018, pp. 1–6 (April 2018)
3. Finsterbusch, M., Richter, C., Rocha, E., Müller, J.A., Hänßgen, K.: A survey of payload-based traffic classification approaches. IEEE Commun. Surv. Tutor. **16**(2), 1135–1156 (2014)
4. Dainotti, A., Pescapé, A., Claffy, K.C.: Issues and future directions in traffic classification. IEEE Netw. **26**(1), 35–40 (2012)
5. Adibi, S.: Traffic classification – packet-, flow-, and application-based approaches. Int. J. Adv. Comput. Sci. Appl. **1**, 6–15 (2010)
6. Haraty, R.A., Zantout, B.: The TOR data communication system: a survey. In: Proceedings - International Symposium on Computers and Communications Workshops, pp. 1–6 (2014)
7. Dingledine, R., Mathewson, N.: Design of a Blocking-resistant Anonymity System, pp. 1–24. svn.torproject.org (2006)

8. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proceedings of the 13th Conference on USENIX Security Symposium, SSYM 2004, USA, vol. 13, p. 21. USENIX Association (2004)
9. Pistunovich, V.I.: Tor: the second-generation Onion router. Sov. At. Energy **46**(4), 337 (2005)
10. Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource routing attacks against Tor, p. 11 (2007)
11. Montieri, A., Ciuonzo, D., Member, S., Aceto, G.: Anonymity services Tor, I2P, JonDonym: classifying in the dark (February 2018)
12. Koch, R., Golling, M., Rodosek, G.D.: How anonymous is the Tor network? A long-term black-box investigation. Computer **49**(3), 42–49 (2016)
13. Diaz, J., Arroyo, D., Rodriguez, F.B.: Fair and accountable anonymity for the Tor network. In: ICETE, vol. 4, pp. 560–565 (2017)
14. Granerud, A.O.: Identifying TLS Abnormalities in Tor. Information Security (2010)
15. Stone, J.A., Saxena, N., Dogan, H.: Systematic analysis: resistance to traffic analysis attacks in Tor system for critical infrastructures. In: Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018, pp. 2832–2837 (2019)
16. Amann, J., Sommer, R.: Exploring Tor's activity through long-term passive TLS traffic measurement. In: Karagiannis, T., Dimitropoulos, X. (eds.) PAM 2016. LNCS, vol. 9631, pp. 3–15. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30505-9_1
17. Ling, Z., Luo, J., Yu, W., Yang, M., Fu, X.: Tor bridge discovery: extensive analysis and large-scale empirical evaluation. IEEE Trans. Parallel Distrib. Syst. **26**(7), 1887–1899 (2015)