Detection and Analysis of Tor Onion Services

Martin Steinebach[1,*], Marcel Schäfer[2], Alexander Karakuz and Katharina Brandl[1]

[1]*Fraunhofer SIT, Germany*
[2]*Fraunhofer USA CESE*
E-mail: steinebach@sit.fraunhofer.de
[*]*Corresponding Author*

Received 28 November 2019; Accepted 28 November 2019;
Publication 23 January 2020

Abstract

Tor onion services can be accessed and hosted anonymously on the Tor network. We analyze the protocols, software types, popularity and uptime of these services by collecting a large amount of *.onion* addresses. Websites are crawled and clustered based on their respective language. In order to also determine the amount of unique websites a de-duplication approach is implemented. To achieve this, we introduce a modular system for the real-time detection and analysis of onion services. Address resolution of onion services is realized via descriptors that are published to and requested from servers on the Tor network that volunteer for this task. We place a set of 20 volunteer servers on the Tor network in order to collect .onion addresses. The analysis of the collected data and its comparison to previous research provides new insights into the current state of Tor onion services and their development. The service scans show a vast variety of protocols with a significant increase in the popularity of anonymous mail servers and Bitcoin clients since 2013. The popularity analysis shows that the majority of Tor client requests is performed only for a small subset of addresses. The overall data reveals further that a large amount of permanent services provide no actual content for Tor users. A significant part consists instead of bots, services offered via multiple domains, or duplicated websites for phishing

Journal of Cyber Security and Mobility, Vol. 9_1, 141–174.
doi: 10.13052/jcsm2245-1439.915

attacks. The total amount of onion services is thus significantly smaller than current statistics suggest.

Keywords: Tor, Darknet, onion services, analysis.

1 Introduction

In recent years, the demand for online privacy has become an increasingly important topic. This is evident both at the political level and in everyday life. New regulations such as The General Data Protection Regulation in the EU [17] or the California Consumer Privacy Act in the USA [5] and an increasing range of privacy friendly online services try to give individuals more control over their personal data and their privacy. This trend is also reflected in the growing number of users that rely on online services as the search engine DuckDuckGo or the anonymisation network Tor, to anonymise their personal data. Over the past four years, DuckDuckGo has registered an annual increase of one to three billion searches [8].

The Tor network, with a total of almost two million daily users [29], was able to establish itself as an important tool for bypassing government surveillance and commercial data collection. Tor is the combination of an open network and open source software that protects its users against online surveillance and traffic analysis. An increasingly popular feature of the Tor network are onion services. These are online applications that, unlike common Internet services, can be both visited and hosted anonymously. They are used in very diverse ways. Organizations and news outlets like The New York Times or The Guardian use them to communicate anonymously with whistle-blowers [20]. Others circumvent censorship or exchange information in countries with strong government surveillance [13]. And again other users utilize them to offer or buy illicit content and goods [4].

The Tor Project, Inc., the organization primarily responsible for maintaining the Tor software, provides statistics in the form of extrapolated and aggregated meta data on the development of the Tor network. These statistics include data like the estimated amount of users or average traffic throughput since the end of 2014. The total amount of .onion addresses (addresses used to identify onion services – similar to Internet URL addresses) has been steadily growing over the past four years. Their estimated amount has quadrupled from 26,878 on 2015-01-01 to a total of 112,628 on 2018-11-15. The same increase can be seen in the onion service related traffic. While on 2015-01-01 the estimated total throughput of data was 0.52 Gbit/s, it

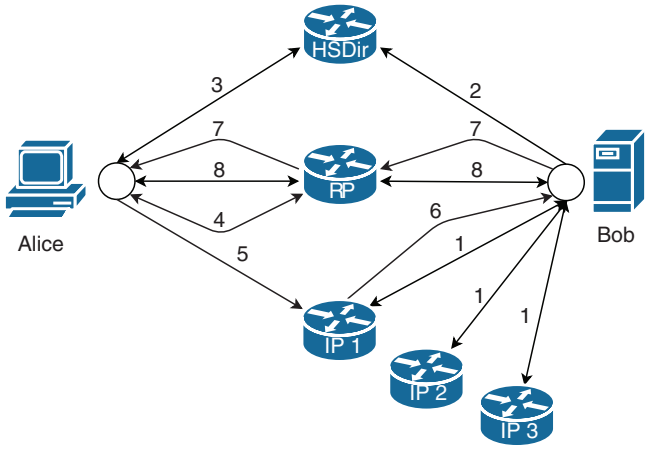

Figure 1 Tor hidden services are announced and reached by a multi-step protocol.

amounted to 1.37 Gbit/s on 2018-11-15. With this huge increment and given the dynamic and anonymous nature of the Tor network, it is hard to picture an accurate state of the network based on those old results. It is rather speculating than determining what kind of applications are hosted, how popular they are or what content they provide.

1.1 Hidden Services

The setup and communication procedures of onion services are documented in [25] (a recent update is availab le in [26]) and can be divided into several steps. For illustrative purposes the protocol is explained based on a communication between the two parties Alice and Bob. Alice represents a client using the onion service whilst Bob represents the provider.

The overall procedure consists of three main phases. In the first phase (Creating an onion service identity) Bob's onion proxy generates all necessary files and a descriptor, containing the contact information with which his service can be reached. During the second phase (Publication of the onion service) this descriptor is published to the responsible Tor relays. In the final phase (Connecting to the onion service) the descriptor is retrieved by the connecting party and subsequently used to establish an anonymous connection. The following exemplification describes the individual steps of each phase in more detail. The numeration of these steps corresponds to the numbers depicted in Figure 1.

In the following, we show the communication between Alice's and Bob's onion proxies OP_A and OP_B. Before the onion service can go online for the first time, Bob's onion proxy OP_B generates an asymmetric key pair. i.e. a private key SK_B and its corresponding public key PK_B. The private key SK_B will be used to sign published data. The public key PK_B is used to verify Bob's signatures and to generate the permanent identifier under which the service can be reached. This identifier is generated by hashing PK_B using SHA1, truncating it after the first 80 bits and encoding the result using Base32. Afterwards a ".onion" suffix is attached to the permanent identifier. This results in what is referred to as the .onion address of the service. This address is used in a similar way as common URLs on the Internet but can only be resolved by the Tor Browser.

Step 1: Establishing introduction circuits
Bob's onion proxy OP_B chooses up to 10 relays from the consensus which will be used as introduction points (IP). Introduction points are relays that serve as the first contact point for the onion services. OP_B creates one new RSA key pair – called service keys – for each introduction point and sends the keys to each relay. The introduction points then reply with a confirmation message and terminate all other circuits, associated with the received service key. This way only one circuit between the introduction point and the onion service can exist. Introduction circuits can only ever be used for onion service introductions.

Step 2: Publishing the Hidden Service descriptor
The next step begins with OP_B generating two HS descriptors. The descriptors contain all information Alice needs in order to establish an anonymous connection to the onion service. Once the descriptors are generated they are published. For each descriptor, OP_B chooses the Hidden Service Directory (HSDir) relay with a relay fingerprint closest to the descriptor-id and the two subsequent neighbors in the hash ring. The relays receive these descriptors via a HTTP-POST requests over Tor circuits. OP_B will publish a new v2 descriptor once every hour or whenever its content changes.

Step 3: Alice's onion proxy OP_A queries the HS descriptor
Alice receives the .onion address from Bob. OP_A uses the address to calculate the current descriptor-ids of the service. OP_A determines the currently responsible HS directories and requests the descriptor from them. The HTTP-GET request is performed to one random server out of the set of six. If the request fails, a different server is queried. The directory responds with the

most recent descriptor published by Bob. Once received, the descriptor is validated. The public key can be validated using the .onion address and the signature can in turn be verified via the public key. Finally the descriptor is stored in Alice's cache.

Step 4: OP_A establishes a rendezvous point
OP_A chooses a random relay from the consensus and establishes a rendezvous circuit by sending it a data package containing a rendezvous cookie. This cookie is an arbitrary 20 byte long value and is associated with the established rendezvous circuit. The rendezvous point responds with a confirmation message.

Step 5: OP_A contacts one of OP_B's introduction points
OP_A chooses one random introduction point and sends that relay an introduction message. This cell contains, among other necessary data, the respective service key, information about the previously established rendezvous point, the first part of a Diffie-Hellman key exchange handshake and the rendezvous cookie.

Step 6: The introduction point relays the rendezvous points to OP_B
The relay successfully matches the service key to an introduction circuit and sends a confirmation message to OP_B forwarding the rendezvous information.

Step 7: OP_B and OP_A establish a connection via the rendezvous point
OP_B creates a circuit to the rendezvous point using the rendezvous cookie and the Deffie-Hellman key exchange response. The rendezvous point forwards OP_B's message to OP_A who also creates the shared secret using their Deffie-Hellman key exchange.

Communication
OP_A opens a TCP-stream to the rendezvous relay. Alice and Bob can exchange data bidirectionally using the rendezvous point as a middle point to relay information. Alice and Bob only know the rendezvous point and are connected to it via a Tor circuit. As a result, both parties remain anonymous.

1.2 Objectives

We aim to determine the types of onion services offered on the Tor network, their respective popularity and uptime. The detected protocols and

applications are clustered to illustrate the variations of hosted services. The popularity data depicts what services are most in demand. Aggregated uptime durations show whether they are hosted permanently or if their use is predominantly short term. A more specific analysis is conducted on websites. At first, they are clustered by their language. This shows whether web content is offered in other languages than English and in what proportion. Second, the collection is filtered for duplicates. This will determine the amount of unique websites.

2 State of the Art

This chapter provides an overview over the past research on the content of onion services and their popularity. The existing data is also evaluated.

2.1 State of the Art Research on Onion Service Usage

In [2], Biryukov et al. demonstrated that it was possible to collect a majority of currently published .onion addresses using a technique they named 'shadowing'. This method takes advantage of an old artefact in the Tor source code. Normally, the directory authorities would include only up to two Tor relays per IP in the consensus. Both relays can then be reached by the combination of their shared IP and a pre-configured port. However, if a server hosted more than just two Tor relays, all of them would still be constantly measured by the directory authorities. The additional relays would obtain flags as if they were part of the Tor network but not be published on the consensus. If one of the two published relays were to go offline, one of the not published relays would be incorporated instead. This relay would then instantly have all the relay flags as if it was part of the consensus all along. These concealed relays were named *shadow relays*. The researchers brute-forced the RSA identity keys of 1,392 relays, so that the resulting fingerprints would place them evenly distributed over the complete Hidden Service Directory (HSDir) hash ring. Once every relay acquired the HSDir flag they gradually took down two relays of each IP every hour and fetched the received descriptors from their memory. This way they were able to traverse the complete Distributed Hash Table twice in 24 hours with just 58 virtual instances. Since every onion service should publish its descriptor at least once every hour, an iterative hourly change in relays would provide descriptors for most onion services running for 24 hours as well as an additional amount of shorter living services. The result was a collection of 39,824 unique .onion addresses.

Three months later Alex Biryukov et al. published their results of a content and popularity analysis on the collected onion services [3]. Out of the 39,824 .onion addresses 24,511 were still accessible during the time of the conducted port scans. The analysis showed a total of 22,007 open ports. The results of the port scan showed that a significant majority of onion services belonged to a botnet called Skynet. The second most popular services were web based. The subsequent crawling of these websites yielded a total of 1,813 classifiable English web pages. The analysis was performed for English texts only (84% of the web pages). Each of the other 16 languages accounted for less then 3% of all websites. The total sum was 432 non-English websites. Web services were the only ones classified by topic. Platforms for drugs and pornographic content were the most frequent. In total, the amount of illicit content made up roughly half of all web pages (category services included since it contained e.g. offers for money laundering or hiring of thieves and hitmen). The two most frequent non-illicit topics were politics and anonymity. According to [3], the politics category included topics like corruption, repressions, violations of human rights and freedom of speech. The anonymity category were websites offering guidelines and discussions on anonymous services and anonymity in general. The researchers also tracked client requests for specific descriptors. During the data collection a total of 1,031,176 requests for 29,123 unique descriptors were logged. 80% of the requests were, however, for descriptors that were never published. The total amount of successfully resolved addresses was 3,140. The majority of requests were related to either the botnet 'Goldnet', the botnet 'Skynet', the drug market 'Silk Road' or pornographic content.

These results also support the findings of [9] which indicated a large centralized botnet using the Tor network to hide the location of its CC servers. 'Skynet' was a botnet created through a modified version of the Zeus Trojan horse malware package. It was used for a variety of purposes such as DDOS attacks or Bitcoin mining. For this purpose, the malware additionally contained the Tor software package, mining tools as well as various software libraries. The individual bots and the CC center hosted onion services on the Tor network and used them to communicate via an IRC protocol. This setup helped to reduce the traceability of the botnet. Another botnet, labeled 'Mevade', was discovered in August of 2013, when the amount of Tor users suddenly spiked from roughly 800,000 to 5,000,000 in the time period of one month [22].

A second study was carried out by Owen and Savage in 2015 [18]. The researchers operated 40 relays on the Tor network and passively collected

descriptors for a duration of 6 months. The collected amount of unique addresses was 80,000, with 15% still online for the subsequent analysis. The researchers estimated an average uptime rate based on whether services republished their descriptor at later points in time. A wide range port scan was not documented and the evaluation focused mainly on a topic classification for the detected websites. The popularity of individual services was measured by their descriptor requests. 0.6% of all descriptors were partially encrypted with a descriptor cookie.

Another study was conducted by Daniel Moore and Thomas Rid in the year 2016 on the usage of onion services. Moore and Rid [14] describes an in-depth scan of onion service websites. The researchers used a website crawler specifically designed to crawl web based onion services. The resulting HTML-data was classified using Support-Vector Machines SVM – a supervised learning model used to categorize text into topics based on a pre-classified training set [30]. The seed, used as basis for the crawler, was taken from ahmia.fi and onion.city. The two search engines provided 5,615 unique web based .onion addresses. Multimedia files such as images or videos where excluded from the analysis due to possible legal complications. The crawler also followed links on the analyzed domains if these linked to another onion service. The final result where a total amount of 5,205 crawled websites, out of which 2,723 were successfully classified into 12 categories. It is noteworthy, that the category finance was compromised mainly of money laundering, illegal credit card trade and counterfeit currency.

2.2 Evaluation of the Available Data on Onion Service Usage

In [3] is shown that in the beginning of 2013, a predominant amount of onion services consisted of bots taking advantage of the anonymity provided by the Tor network. While a total amount of 39,824 onion services suggested a decently sized network of services, the analysis revealed that just a fraction actually provided content for Tor clients. After removing websites with less than 20 words and those which solely contained the default Torhost.onion hosting service page, only 1,813 English web pages remained. Even including the most frequent chat applications and not analysed non-English websites, the services amounted to less than three thousand. Also, while the popularity analysis revealed a total of 1,031,176 requests in just 24 hours, only 20% of them referred to services that were also online. This suggests a high amount of automatised queries by bots and other scanners, instead of actual Tor users. The study [18] was based on a 6 months observation. The data enabled the

researchers to determine an estimated uptime for the collected services. Only 15% of all services existed for a duration of 6 months suggesting a high churn rate. Their popularity analysis depicted roughly the same results as in [3], with the majority of descriptor requests being either for pornographic content, drug markets or botnet related onion services.

Both studies did not consider the issue of duplicate web services. Some websites can be represented multiple times in the collection of the crawled HTML-data. Due to their cryptic appearance .onion addresses might even be more prone to phishing attacks or they could be reachable through multiple .onion domains. Such duplicates may lead to an over-representation of specific services in the data set. The data might also no longer provide an accurate portrayal. As stated above, the total amount of onion services has almost quadrupled since the first study was conducted. The botnets might no longer be active and the nature of the dominant services could have changed.

2.3 Collection of .onion Addresses

There are several ways to collect .onion addresses. The naive approach might be to brute-force all possible .onion addresses and probe the results in order to determine those currently used. An address is 16 characters long with 32 possible values for each character. The total amount of possible unique variations is 32^{16}. This solution is hence not feasible, especially as the next generation ("v3") address length will be 56 characters (e.g. [24]).

Another approach is collecting publicly available .onion addresses. Several previously described onion services index a few thousand addresses that were published online or found by crawling. As described in 2.2, these addresses are, however, just a fraction of the total amount. They are also, for the most part, limited to web domains. This approach would not provide a sufficiently diverse representation. This would also skip over possible botnets.

A third approach is to host relays with a hidden service directory (HSD) flag and use those to collect the data set. Every hidden service descriptor HS is published to two pseudo-randomly selected relays and to their two successors in the hash ring. In order to collect all published HS descriptors at any given time, one would have to place a new relay in every second gap of the directory hash ring. According to [23], the hash ring consisted on average of 3,605 nodes during the months April to September 2018. This means that at least a total of 1,803 relays would have to be hosted, requiring 902 individual IPs. Whilst this approach would produce the best result, it also requires a high amount of resources.

3 Concept

This section conceptualizes an approach for the real-time detection and examination of onion services. The approach is based on the objective to collect and evaluate data on their software type, popularity, uptime and, for web services, content.

3.1 Relay-based Onion Address Collection

The quantity of the collected addresses should be high enough to be considered representative for the collective as a whole. As discussed in the previous section, known approaches are either incomplete or resource-intensive.

Instead, in this paper we propose a slightly different approach: We collected descriptors over a longer period of time by incorporating only a small subset of relays. Given a sufficiently long time span these relays are able to collect most onion services that are permanently on the network as well as a representative subset of short-lived ones.

The amount of relays and the needed time span can be calculated using the geometric distribution model with each change of the descriptor-id as a single Bernoulli trial. For this model to apply, three assumptions must hold true. First, the sequence of events have to be independent trials. This assumption applies, since the calculation of a new descriptor-id is independent of the previous values. Second, the occurrence is a binomial trial. Each daily change of the descriptor-id can be defined as either a successful or unsuccessful outcome. The successful being that at least one of the recording relays received the descriptor of the permanent onion service. Therefore, the second assumption applies as well. The third assumption is that the probability of success remains the same for every trial. This proves to be problematic. There is no way of predicting the exact probability of future publications since the amount of total relays fluctuates. The fluctuation in the past months has, however, been marginal with a maximal increase by 170 and a maximal decrease by 41 relays (considering the total number of relays from 2018-04-01 until 2018-09-01, taking the first day of each month [23]). We will hence assume a constant probability based on the average amount of relays in the period from 2018-04-01 to 2018-09-30. The probability of a single onion service publishing a descriptor to one of the recording relays, at least once during a given time period *n*, can then be calculated as:

$$P(X \geq 1) = 1 - (1 - p)^n$$

Every descriptor is published to two relays and their two successors respectively. If we assume that none of the recording relays are ever in the same receiver group, the adjusted formula can be written as:

$$P(X \geq 1) = 1 - (1 - p)^{2n}$$

Consequentially, p can be determined by the amount of needed relays in relation to the total amount of hidden service directory relays.

$$P(X \geq 1) = 1 - \left(1 - \frac{3r}{a}\right)^{2n}$$

where:
p: The probability of receiving the descriptor on a descriptor-id change
n: The amount of descriptor-id changes (days)
r: The amount of owned HSD relays
a: The total amount of HSD relays on the network.

3.2 Detection of the Underlying Services

In order to classify the onion services by type it is necessary to detect the applications running on them. Hence, the next step is a port scan over each service. An open port, however, only implies the possible type of application listening on this port. It is e.g. possible that the host runs a web service on a port usually used for IRC applications. Because of this, the detection of an open port is followed by a best effort detection of the underlying service and operating system. The studies described in 2.1 have also shown another important aspect that has to be taken into account: Some services have a short life span. This means that it is important to conduct the analysis as fast as possible after their detection. Hence, the implemented system scans each service soon after its publication.

3.3 Evaluation and Categorization

The evaluation provides insight into the following aspects of the detected onion services:

Type of service
All recorded addresses and the respective services found during the port scan were persisted in a database. The data fields indicated the type of service, the operating system and additional information on the used software for each address.

Amount of services protected by descriptor cookies
Onion services using the basic client authorization method cannot be tracked, even when the HSD descriptor is known. A connection cannot be established since the introduction points are encrypted. The key length of such a cookie can vary but Tor Project recommends a length of 128 bits. To brute force this secret would, for one, be exceptionally time consuming and secondly compromise their privacy. For this reason, services protected by this method, had to be excluded from the analysis. Still, by checking whether the introduction points of each published descriptor are encrypted, we estimated the percentage of services relying on this technique.

Uptime
For each collected *.onion* address the lifetime of the underlying service was tracked. This statistic provides insight into the persistence of onion services. It shows whether the majority of applications are used for short term data exchanges or whether they provide long term services for Tor clients.

Popularity
All requests for HSD descriptors were tracked for both v2 and v3 services. While this form of popularity tracking does not differentiate between real Tor users and automatic requests (e.g. by bots, scanners or crawlers) it showed how often specific services are requested on the network. Also, the logged requests were distinguished between such that failed to be resolved and those that were successful. The most popular services were examined more closely. Finally, the popularity of v2 and v3 services was compared to show how well the new protocol was adopted by Tor's users.

Language of each web service and duplicates
Each detected website was additionally crawled and the underlying HTML-data persisted for further analysis. Due to the possibility of illicit content in the form of images or video data the crawler refused downloading any multimedia content and processed solely HTML-text. Also, it followed all links to other *.onion* domains. Each web domain was categorized by the main language and similar websites were filtered from the collection.

4 Implementation

This section describes the architectural decisions made during the implementation.

4.1 Resources

The hosted relays are only capable to capture a fraction of the Tor Distributed Hash Table (DHT). To collect a representative amount of addresses, we had to record for a sufficient duration. This time span had to be long enough for most permanent onion services to publish at least once to one of those relays. For this purpose, we determined the desired probability of receiving a descriptor from a permanent service to be at least 90% or an equivalent of 90 days. Consequently to the analysis in 2.3, the average amount of HDS relays on the network was estimated to be 3,605 for the duration of the data collection. Inserting these values and a publication probability threshold of 95% into the formula defined in 2.3 we were able to calculate the required resources of 20 relays. Since two relays can be hosted per IP, the amount of needed servers and IPs was reduced to 10.

The data collection and analysis was performed on separate systems. An additional system was used for each server. Each performed the analysis solely for its respective server. This separation allowed for a fast and dispensed evaluation and also helped to prevent race conditions. The resulting data is persisted in a remote central database.

4.2 Architecture

The overall architectural structure and the interactions of the individual systems are illustrated in Figure 2. A set of 10 virtual machines hosts a total of 20 relays on the Tor network. The relays run a modified version of the most recent Tor software package (0.3.4.9). The modifications trigger an event whenever a descriptor is published to the relay or whenever one is requested by a Tor client. The respective meta data of each event can then be read at the control port of the Tor process. Scripts, subscribed to these events, archive the publications and requests to respective log files.

Additionally, the data is also processed and persisted in a remote database. Analysis tasks are performed on an additional set of 10 VMs of which each is responsible for two relays. They retrieve and process the data based on predefined relay fingerprints. The overall process is controlled through a remote workstation which uploads, starts and stops the used scripts and performs other administrative tasks. For authentication purposes and to prevent the transmission of data in plain text a simple PKI was set up.

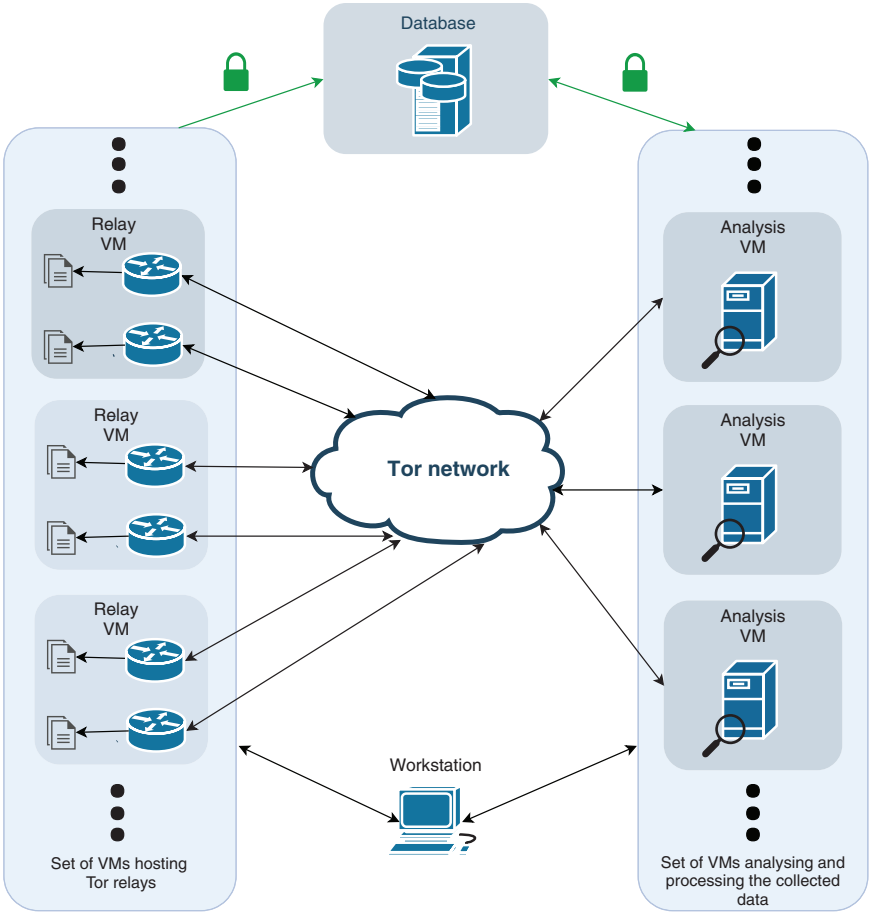

Figure 2 Architecture overview.

4.3 Harvesting Hidden Service Descriptors

Tor relays with the HSD flag are placed in the DHT according to the SHA1 value of their public RSA identity key. In order to achieve the best possible coverage of the DHT and to prevent the hosted relays from being in the same receiver group for individual descriptors, new RSA key pairs were brute forced. The new keys were chosen in a manner that would place the relays evenly spread over the hash ring. Once the necessary flags were acquired, the modified Tor software triggers events whenever a descriptor was published to the node. Instead of directly persisting the published data to the database, it

was written to a log file. This redundancy provided an additional backup of the collected data and ensured the continuity of the collection process. The individual log files consisted of the following data.

- Received v2 descriptors
- Received v3 descriptors
- Resolved v2 descriptor requests with indicator if successful
- Resolved v3 descriptor requests with indicator if successful

The log file for v2-descriptors is constantly pooled for new publications. An additional offset file markes the last successfully parsed entry. Since the descriptors do not include the *.onion* address of the respective service, it is calculated by hashing the enclosed public RSA key, taking the first 10 bytes of the result and appending the ".onion" suffix. The usage of descriptor cookies is detected by inspecting the introduction points field within the descriptor and checking whether the information is encrypted.

4.4 Logging the Amount of Descriptor Request

The popularity of individual services was determined by evaluating the requests for each descriptor. Since Tor clients use the descriptor-id, instead of the .onion address, the underlying address cannot be determined by the request alone. To map the requests onto the respective address, the *onion_descriptor_history* table is queried for the corresponding addresses. The log files were parsed as a whole at the end of the data collection. Since the requests could always be compared with every descriptor-id logged during this time, no data was lost. The database entries for failed requests were aggregated by their descriptor-id.

The log files for v3-descriptors were also parsed as a whole at the end of the data collection phase. Since v3-descriptors are encrypted they cannot be mapped to their respective .onion address without knowing it beforehand. The collected data can be used in two ways. For one, the total amount of failed and successful requests can still be determined. Secondly, v3-addresses found during the crawling of v2-services can retrospectively be linked to the request logs. This is realized by comparing the blinded public keys, each collected v3-service has during the analysis phase, with the logged requests.

4.5 Logging the Uptime of Onion Services

The uptime of an onion service can be determined by routinely querying each service directly via a connection to one of its ports. Whether the port is open

or closed is irrelevant since a received answer does already indicate that the service itself is running.

A query during the first 24 hours is not necessary, since the service will publish the descriptors itself. However, if the publication stops within this first time frame, a query will be performed every 3 hours. This allows for a window of 3 failed publications. After this first period, queries are performed once on a daily basis for the first week. After one week, the scans occur every 7 days and after one month every 4 weeks. In order to compensate for short outages of the service, this routine repeats itself on every status change of the service. This means that whenever a service goes from online to offline, or vice versa, the routine resets to the first phase for this specific service.

These requests might, however, influence the popularity analysis. If the queries were performed at the collecting relays, these would in turn log an increased amount of potential visits. This is prevented by calculating the responsible relays for each descriptor-id beforehand and ignoring the recording relays during the query.

4.6 Port and Software Detection

The classification of the logged onion services was determined by their open ports and the underlying software. In order to collect this data, a port scan routine with a simultaneous software detection was implemented.

By default, onion services slow down port scans by closing rendezvous point circuits whenever an unrecognized port is accessed [28]. Hence, every detection of a closed port results in a repeated setup routine for a new rendezvous point circuit. This means that the scan time per service has to be taken into account. Test runs with common port scanners like *Nmap*, *Ncat* and *Netcat* revealed that, while Ncat performed the fastest for single service scans, Nmap outperformed all other scanners when a set of addresses was used. Additionally, Nmap allows to simultaneously check detected ports for more than 2,000 well-known services [16].

The procedure is an incremental scan of each service. Instead of scanning the complete port range, only an increasing subset of ports is scanned until at least one open port is detected. Since web applications appear to be the most frequent services, the port scanner starts with port 80 and 443. Subsequently, a variety of common Tor applications, such as TorChat, Jabber, common IRC and RPC ports, Torrent tracker and ports known to be used by the Skynet and Trickbot botnets, are examined. This is followed by an additional scan of the 100 most commonly used ports (as determined by Nmap's list of well

known ports [15]). Finally, if no open port is found, the scan continues with the 1,000 most common ports. This set of ports does include some of the previously scanned ones.

4.7 Web Services

Subsequently to the port scan, each onion service was crawled for which an open port with the HTTP was detected. The HTML-text data was collected and persisted for further analysis. Once the data collection was completed, the text was classified by its language and type of content.

The crawler was specifically designed to crawl onion web services and provides the option to exclude multimedia data due to possible legal complications. Multimedia content like pictures or videos were not downloaded and the analysis of the web services was limited to the returned HTML-texts. Web services detected through links were also crawled, provided they linked to another onion service. The crawler was modified so that it can run on multiple VM's in parallel without performing crawls multiple times on the same service.

Subsequent to the crawling, the HTML-data was filtered and clustered by different procedures. The first procedure filtered out web services that returned solely HTTP-error codes and marked them accordingly for later evaluation. The second procedure clustered duplicate websites. Some websites can be offered via multiple addresses or be cloned to carry out phishing attacks.

4.8 Classification of Languages

Additionally to the previous clustering methods, the collected websites were also classified by their main language.

To achieve a precise categorization, the documents are classified based on a voting poll of three natural language processing applications. The respective applications are *fastText* (a NLP library created by Facebook's AI Research lab), *polyglot* (a natural language pipeline and language detector using Google's C++ library Compact Language Detector 2) and *langdetect* (a python based port of Google's language-detection library). Each tool votes on the three most likely languages for the text – also stating the estimated probability. The main language is determined by a majority vote of two or, if no majority can be achieved, by the overall most probable language.

5 Results

In total, a sum of 318,787 unique descriptors were published to the relays. These resolved to 173,190 unique *.onion* addresses. The discrepancy between the amount of descriptors and *.onion* addresses has two causes. First, different relays were responsible for the same *.onion* address at different points in time. Second, contrary to the Tor protocol, some services published multiple descriptors to the relays whether they were currently responsible for them or not. The detection of potential honeypot services requires an active querying process of the published addresses, which was done only in a shorter amount of time for technical and policy reasons. However, those services could still be retrospectively identified. This was done by comparing the collected data with archived consensus documents. Each publication was then checked on whether it was published to the correct position. In total, 850 descriptors were published to relays which were not responsible for the received descriptor-id. These publications were performed by only 216 onion services. All of these services are HTTP-based but appear to have nothing else in common.

The crawler provided an additional amount of 3,086 addresses. Out of all detected services, only 6,186 used a descriptor cookie. That is, only 3.57% of the collected services used the basic client authorization feature to encrypt their introduction points. Since the cookie prevents any unauthorized access to these services, 170,090 remained to be analyzed.

5.1 Services

Out of the 170,090 accessible onion services (collected by the relays and the crawler) 82,145 were scanned to different extends. The discrepancy of 52% is the result of the interruption in the data collection process and the subsequent passive monitoring during which the missing services went offline again. For 60,036, i.e. 73% of those services, at least one open port was detected. Because most services of the second data collection had to be scanned collectively at the end, there was not enough time left to scan the complete port range of all services. Toward the end of the data evaluation, 9,966 services with no detected open ports were still online. The achieved port range coverage for these services was 45%.

5.1.1 Distribution of service types

The collection of 60,036 services yielded a total of 65,987 open ports. These belong to 219 unique protocols on 1,370 unique port numbers.

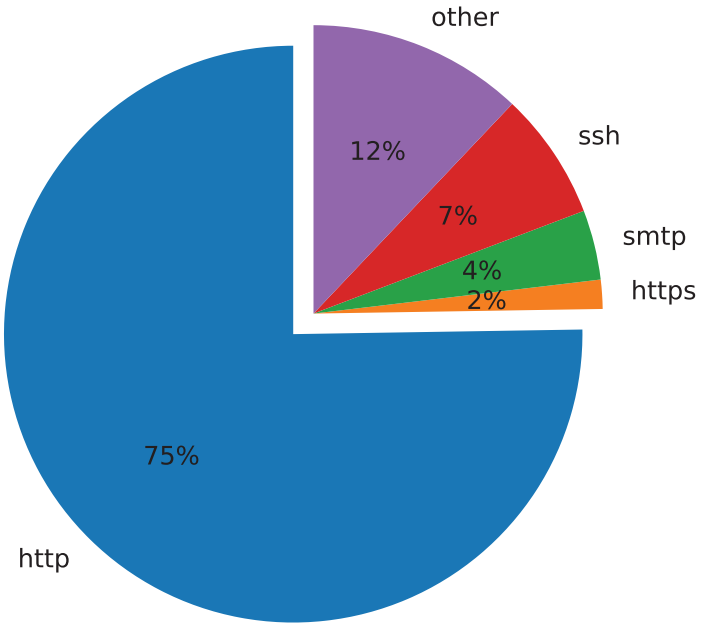

Figure 3 Distribution of all protocols.

The results show that a majority of 49,659 open ports, are used for the HTTP protocol. This is followed by the SSH protocol and the SMTP. 92% of all HTTP-services were found on port 80. 77% of SSH-services accept connections at port 22 and 95% of SMTP-services on port 25. Figures 3 and 4 illustrate this distribution. The remaining of the 10 most common protocols follow the same pattern of predominantly complying with the standard port allocations. While HTTP and SSH based services were also the most frequently detected protocols in [3], the SMTP and Bitcoin protocols were not. This shows that a change in service usage occurred between the years 2013 and 2018. Anonymous Email servers and Bitcoin clients have become significantly more popular.

Only 1,076 of the detected onion services use HTTPS. Subsequent to the crawling, these services were queried for the respective certificates. 843 of them were still reachable. 333, i.e. 39,5%, of the TLS certificates were self signed. Of the not self signed certificates, the most commonly used CA was the Let's Encrypt Authority with 276 (32.7%) issued certificates followed by the COMODO RSA Domain Validation Secure Server CA with 20. Additionally, the trust chain was also examined.

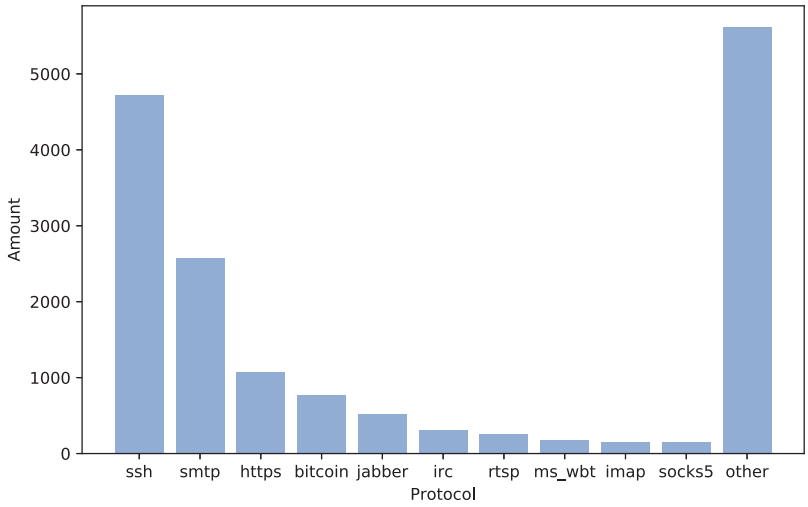

Figure 4 Top 10 most frequent protocols without http.

5.1.2 Distribution of detected software

For 54,849 (83%) open ports the underlying software was determined. The remaining ports could not be classified because either the software product was not part of the Nmap service database or because a firewall prevented the detection. Table 1 lists the most popular applications for the 5 most frequent protocols (Bitcoin was excluded because no underlying application was determined).

In consistence with the protocol distribution, the majority of detected software products were web servers. The most popular web server is Nginx which is also, together with lighttpd, the software recommended by The Tor Project [27]. Even though Tor Project specifically discourages the use of Apache web servers, as they may be more prone to de-anonymization attacks, it is still the second most popular web server. All of the software products depicted above are open source products with a majority putting also an emphasis on security features.

The port scans were able to detect the version of an application in 11,207 cases. This data shows that onion service hosts, for the most part, tend to not keep them up to date. In fact, many applications are outdated by at least 2 years.

Table 1 Software distribution

	Software	Amount
SSH	OpenSSH	3,931
	Dropbear	257
	MikroTik RouterOS sshd	210
	other	66
SMTP	Postfix	2,506
	Exim	19
	Netqmail	5
	other	9
Jabber	Prosody server	284
	Prosody client	134
	ejabberd	56
	other	31
IRC	InspIRCd	103
	ngircd	31
	UnrealIRCd	25
	other	42
HTTP	Nginx	38,960
	Apache httpd	4,294
	lighttpd	551
	Node.js	503
	other	827

5.1.3 Operating systems

The underlying operating system was detected on 5,074 services. As can be seen in 5, almost all of them are Linux or Unix derivatives with a majority of 84% being either Debian or Ubuntu. Figure 5 provides a graphical illustration of the distribution.

5.2 Service Uptimes

This section evaluates and illustrates the aggregated data on the uptime of the discovered onion services. These are clustered into three groups. The cluster “Permanent” encompasses all services which were online for at least 90% of the time. All services were monitored for at least two months. The cluster “Once” contains those that appeared for one contiguous time span and afterwards never reappeared. The third cluster, “Mixed”, are

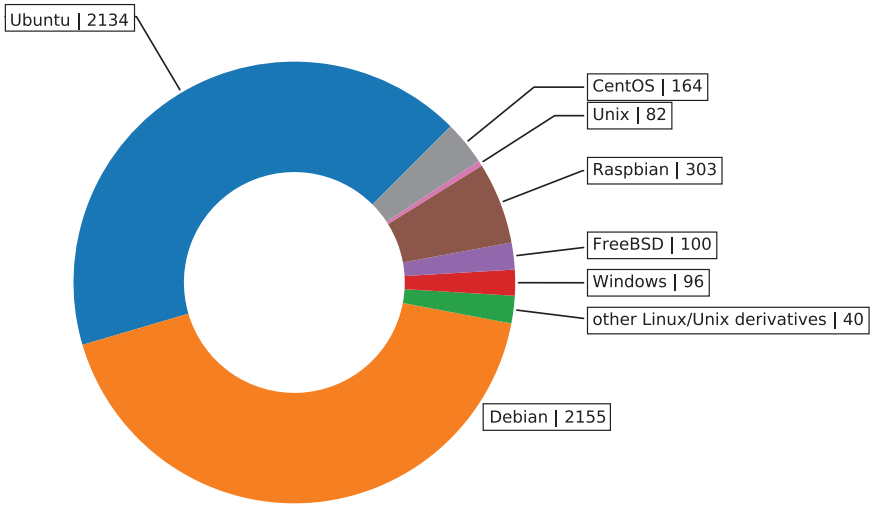

Figure 5 Distribution of operating systems.

services that were periodically on- and offline. The complete amount of evaluated services is 47,439.

The overall distribution of the respective clusters is depicted in Figure 6. To avoid that the relays are banned, the active analysis of the onion services during the last two weeks was conducted with a delay of two to three days after the first publication. 57% of the evaluated services appeared only once and for a fixed time span. The respective time span clusters are illustrated in Figure 7. The data shows that 80% of the services within the "Once" cluster were online for a time span of equal to or less than one week. This is equal to 46% of all evaluated services.

However, this data does not show the ratio of permanent to non-permanent services. It shows solely the distribution of the collected data. The "Once" cluster is over-represented. Permanent services, that published to the relays multiple times, are considered only once while each publication of a short living onion service will lead to an increase of the "Once" cluster. To estimate the actual proportions of permanent to non-permanent services we can expand the cluster by also including those that published a descriptor during the passive recording and were still online for two months after the data collection. By comparing this data with the estimates on total amount of services provided by [23], the real ratio can be approximated. The requirement of an average uptime of over 90% is considered only

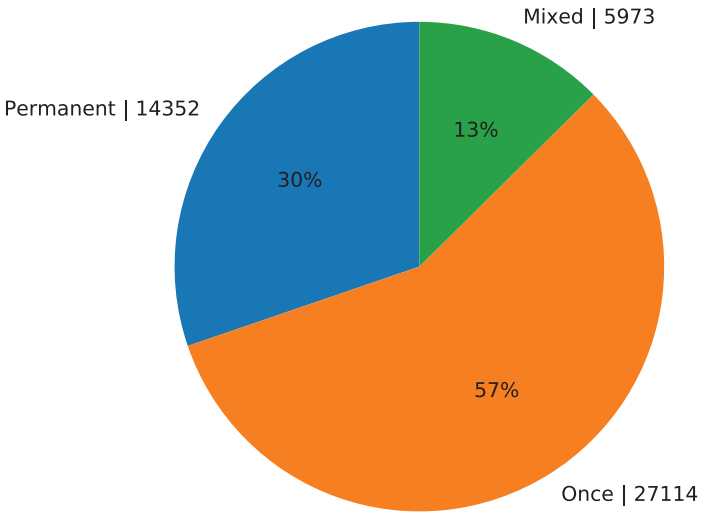

Figure 6 Distribution of all monitored services.

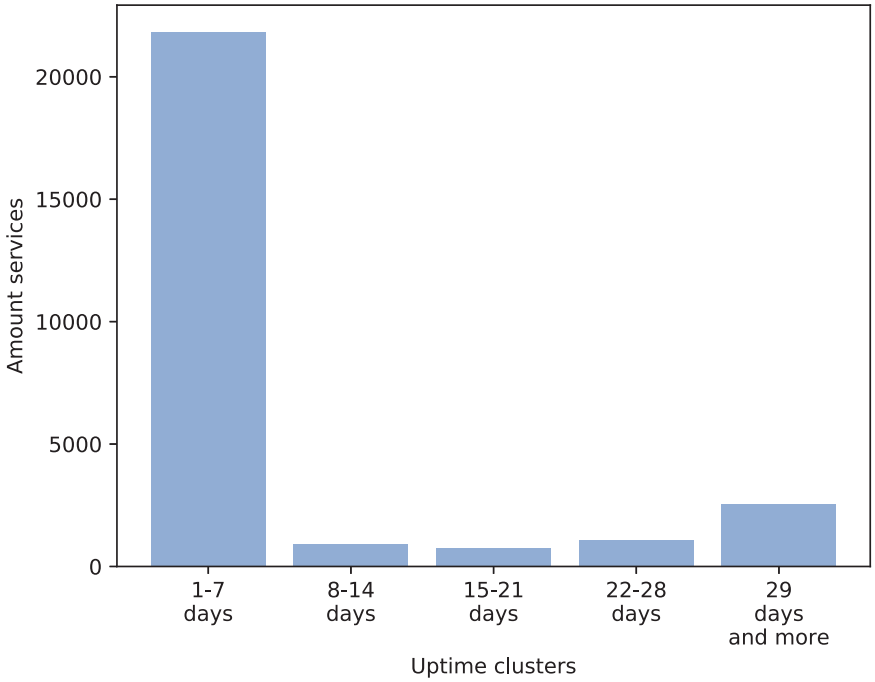

Figure 7 Distribution of the "Once" cluster.

for the subsequent two months period. This results in a total amount of 53,466 permanent services. The Tor Project's extrapolated average amount of total online onion services for the same duration was on average 105.000 services per day. We can hence estimate that roughly half of all services were continuously online for at least a period of 2 months. Over 27,000 of these permanent services belong most likely to a botnet. A difference in the ratio of used protocols between permanent and non-permanent services could not be established.

5.3 Popularity of v2-Services vs. v3-Services

The total amount of descriptor queries for v3-services was 6,493,352. Of these, 2,607,761 were successfully resolved. A total of 1,828 unique v3-descriptors were successfully requested compared to 97,504 unique v2-descriptors. Since the v3-protocol does not include .onion addresses, it is not possible to categorise those addresses by their service type. 371 descriptor-ids were retrospectively assigned to their .onion address. This was done by recalculating the past blinded public keys of all v3-addresses, found during the crawling procedure, and comparing them with the descriptor query records. Of these, the 5 most requested services were all web-based file hosts. The average request rate was 1,426 per descriptor-id. While the amount of successfully resolved v3-queries is less than half of the resolved v2-queries, the total amount of unique v2-descriptors was 53 times higher. This discrepancy indicates that the majority of v3-services are, for the most part, popular services that adapted early to the new protocol.

5.4 Popularity of Cookie Protected Services

A total amount of 6,390 v2-services were determined to be using a descriptor cookie. Out of these, 2,782 services were queried at least once. Collectively, these services were requested 45,542 times with an average request rate of 16.37 requests per service.

5.5 Language Distribution

The following section depicts the language distribution among the web based onion services. All languages are named according to their ISO 639-1 language code. Figure 8 provides an overview of their distribution.

Compared to the analysis in [3] in 2013, almost all languages still constitute for less then 3% of all websites. The sole exception is Russian. In

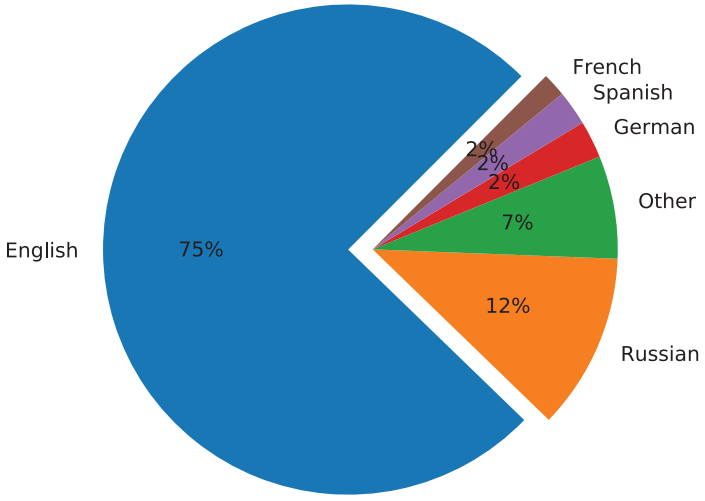

Figure 8 Distribution of main languages used on unique onion service websites.

correlation with the increased use of Tor in Russia, the amount of Russian web services also grew. While the same holds true for most of the other languages, Russian is the only language to outgrow the previous proportions.

5.6 Top 10 Most Requested Onion Services

The 10 services that received the most requests were further analyzed to determine their nature. For each service the most requested descriptor-id was chosen. Four services were disregarded because no information on them was obtainable.

1. **LM Social Server/Backdoor Trojan – 864,505 queries** – The only detected open port on this service has the number 1111, a chat server called LikeMinds social server. This software is no longer available and was integrated into IBM's WebSphere product, using port 1414 [12]. This port is also known to be used by the backdoor remote access Trojans Backdoor.AIMvision [10], Backdoor.Ultor [11], Backdoor.Daodan [21] and W32.Suclove.A@mm [21].
2. **TrickBot – 161,421 queries** – The *.onion* address of this service was associated in [7] with a Trojan called TrickBot. TrickBot is a malware that appeared first in 2016, infecting its targets via phishing attacks disguised as financial institutions. The Trojan consists of multiple modules

and can, depending on the current installation, perform a variety of attacks like collecting emails and credentials, system encryption or even steal Bitcoins from wallets [31]. In September 2018 the National Cyber Security Centre in the UK issued an advisory warning relating to the Trojan. The warning detailed mitigations that organisations and individuals should implement immediately [6]. In order to modify and update its modules, the malware uses a CC server hosted as an onion service. The *.onion* address was associated with the Trojan because it was hard-coded into its binary configuration file.

3. **Wall Street Market – 75,369 queries** – This web based onion service was an online marketplace. It offered a variety of products categorised into drugs, counterfeits, jewellery, carding ware, services, software, hosting, fraud, digital goods and tutorials. It was in the news recently – but long after the end of the evaluation for this paper – for being taken down by a major police investigation, see e.g. [19].
4. **Phishing site of a Tor search engine – 61,086 queries** – This website belongs to a collection of phishing sites that were discontinued sometime toward the end of 2018. The real onion service, which was cloned by this one, is called "not Evil", a search engine in the Tor network.
5. **Hackers Collective – 52,559 queries** – A website claiming to be hosted by a collection of hackers. It offers services like hacking of social media accounts, cooperate databases or DDOS attacks.
6. **Tradizia – 52,111 queries** – A web-based, Russian encyclopedia in the style of Wikipedia.
7. **The Pirate Bay – 49,395 queries** – This service is the Tor web presence of The Pirate Bay. A web based Bittorrent indexer that was started by the Swedish anti-copyright organisation Piratbyrån. Bittorent is a protocol used for peer-to-peer file sharing [1].
8. **Nitrogensports – 43,992 queries** – A web-based betting service. Allows to anonymously bet on sport events or play a variety of gambling games. The wagers are placed in Bitcoins. Interestingly, the *.onion* service redirects the user always directly to a *.eu* domain.
9. **RosPrawosudie – 33,309 queries** – A Russian website created in 2012 for the study of public judicial practice. This free web service offers a collection of past judicial decisions of different jurisdictions. The texts are obtained from official websites blocked on Russian territory by the The Federal Service for Supervision of Communications, Information Technology and Mass Media in Russia – for illegal distribution of personal data [32].

10. **HYDRA Market – 32,474 queries** – This onion service is a Russian web-based market platform. Vendors offer drugs, security services, counterfeits of legal documents, electronic devices, job offers and other goods and services.

5.7 Duplicate Clusters

The duplicate clustering algorithm found 33,317 duplicates divided in 1,021 clusters. Each cluster consists of a collection of duplicates and an average similarity rate based on the determined Jaccard similarity coefficient. Table 2 shows the ten largest duplicate clusters.

The first and largest collection consists of 27,104 duplicates with a similarity rate of 1.0 (100% identical). The only detected port, for all of these services, was port 80. Each service is run on a Nginx web server which hosts a website with just one page. The page displays the logos of the operating systems Linux and FreeBSD. Within the source code of the

Table 2 Ten largest duplicate clusters

Amount	Avg. similarity coefficient	Description
27,104	1.000	Collection of websites displaying the logos of the operating systems Linux and FreeBSD. Have also a hidden text body listing a variety of software and security related skills of a Linux/Unix system administrator.
656	0.999	Collection of former phishing websites. They now display a message of the host explaining the attack.
346	0.998	Default index page of a Nginx web server.
273	0.972	A collection of websites claiming to collect Bitcoins and return a multiplied amount within 24 hours.
252	0.986	Single message saying this website is hosted by Daniel's hosting service.
172	0.902	DreamMarket login page.
128	0.993	Botoshop login page. A Russian service creating customised bot programs.
126	0.925	Default index page of Apache2 Debian web server.
100	0.9874	Default index page of cPanel Inc. showing that the website is no longer reachable.
90	0.865	Index page of torproject.org's developer machines.

website is a small, not displayed text body. The text lists a variety of software and security related skills of a Linux/Unix system administrator and provides contact information, leading to an anonymous Russian Jabber service called "exploit.im".

This layout was found in all 27,104 services. The exact same text can also be found in some Russian forums where users offer IT services. The amount of services also fits a noticeable fluctuation, seen in the extrapolated amount of onion services as provided by [23]. The amount rose drastically 4 times in the past year, while also falling three times in between. The first surge in numbers occurred on 2018-04-25, where the estimated amount increased by 26,455 within 24 hours. Shortly after, the number dropped by 29,899 on 2018-05-08 and rose again on 2018-05-11 by 29,555. This pattern is repeated two more times. Each decrease and increase ranges in between 28,000 and 31,000 .onion addresses. The short time spans and drastic increases strongly indicate an automated process, most likely connected to a botnet. Considering the normal fluctuations and the fact that the amount is only extrapolated, this botnet might be responsible for the observed behaviour.

Similar to the results provided by [18] and [3], this finding supports that the Tor network is used by botnets and malware to hide their location and traffic.

The second biggest cluster consists of 656 duplicates. The average similarity rate is 0.999. All of these websites have just one page with the title "You have been scammed!". The host claims to have been hosting over 800 onion domains on the Tor network with an average of 5.000 hits per day. The descriptor of one of those web services was even requested 61,086 times, as mentioned above. All domains were phishing websites and aimed to collect Bitcoin payments or donations. The host further claims to have received over 200 Bitcoins which led him to retire and reveal the attack. This change occurred roughly at the middle of the crawling process. Since some of the duplicates were, therefore, crawled before and some after the phishing attack stopped, not all of the supposedly 800 duplicates can be identified retrospectively. Still, if the phishing domain was crawled before the reveal and another website linked to these phishing domains afterwards, then both versions of this websites were persisted. In 173 cases both versions of the web domains are available. An evaluation of the duplicates revealed that just in this subset the attacker had cloned 68 different unique services. While most of the duplicates were clones of DreamMarket – an online marketplace for drugs and other illegal goods – they also consisted of a vast variety of

other services like mail, carding and escrow services, search engines, other drug markets, Bitcoin mixers, image hosts and many more.

6 Summary and Further Work

Tor onion services are TCP-based services that can be accessed and hosted anonymously on the Tor network. We provide an overview on protocols, software types, popularity and uptime of these services by collecting a large amount of .onion addresses. Websites are crawled and clustered based on their respective language. The analysis of the collected data and its comparison to previous research provides new insights into the current state of Tor onion services and their development. The service scans show a vast variety of protocols with a significant increase in the popularity of anonymous mail servers and Bitcoin clients since 2013. In correlation with an increased amount of Russian Tor users, the quantity of Russian websites also increased considerably over the past years. The evaluation of the respective service lifetimes reveals a high churn rate for onion services. The popularity analysis shows that the majority of Tor client requests is performed only for a small subset of addresses. The overall data reveals further that a large amount of permanent services provides no actual content for Tor users. Instead, these consist of bots, services offered via multiple domains, or duplicated websites for phishing attacks. The total amount of unique onion services, used by Tor users, is thus smaller than current statistics suggest.

The conducted research can be continued in several areas. First of all, the collected data can be further evaluated. One interesting aspect is the topic classification of the collected HTML-texts. A definitive segmentation of the websites, by their respective topics, would provide an overview of what they are used for. Given the large amount of text data, the classification could be realized using machine learning. First, a manual topic classification would be needed to generate training data. This data can then be used to train a supervised learning algorithm which would classify the remaining texts. The resulting topics can be ranked using the collected popularity data to determine the most popular topics.

A second possible continuation is the in-depth analysis of the found botnets, phishing sites or otherwise fraudulent services. An extensive study on the usage of Tor onion services for malicious purposes could reveal the approaches and methods of attackers. Resulting findings could in turn be used to warn users or for the implementation of counter measures.

Acknowledgment

The joint project PANDA on which this publication is based was funded by the Federal Ministry of Education and Research under the funding codes 13N14355 and 13N14356. The authors are responsible for the content of this publication.

References

[1] The Pirate Bay. The pirate bay – about. https://thepiratebay.org/about, 2019. [Online; As seen on 04 February 2019].
[2] A. Biryukov and Weinmann R. Pustogarov, I. Trawling for tor hidden services: Detection, measurement, deanonymization. *2013 IEEE Symposium on Security and Privacy*, 2013.
[3] A. Biryukov, R. Weinmann, I. Pustogarov and F. Thill. Content and popularity analysis of tor hidden services. 2013.
[4] J. Buxton and T. Bingham. The rise and challenge of dark net drug markets. 2015.
[5] "Legislative Counsel California". California consumer privacy act. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375, 2018. Assembly Bill No. 375.
[6] U. K. National Cyber Security Centre. Advisory: Trickbot banking trojan. https://www.ncsc.gov.uk/alerts/trickbot-banking-trojan, 2018. [Online; As seen on 03 February 2019].
[7] N. Desai. Summer reruns: Threat actors are sticking with malware that works. https://cofense.com/summer-reruns-threat-actors-sticking-malware-works/, 2018. [Online; As seen on 03 February 2019].
[8] DuckDuckGo.com. Duckduckgo traffic. https://duckduckgo.com/traffic, 2019. [Online; As seen on 01 February 2019].
[9] C. Guarnieri and M. Schloesser. Skynet, a tor-powered botnet straight from reddit. https://blog.rapid7.com/2012/12/06/skynet-a-tor-powered-botnet-straight-from-reddit/, 2012. [Online; As seen on 10 November 2018].
[10] K. Hayashi. Backdoor.aimvision. https://www.symantec.com/security-center/writeup/2002-061316-4604-99, 2002. [Online; As seen on 01 February 2019].
[11] D. Knowles. Backdoor.ultor. https://www.symantec.com/security-center/writeup/2002-101713-3321-99, 2002. [Online; As seen on 01 February 2019].

[12] B. Lesser, G. Guilizzoni, J. Lott, J. Reinhardt and R. Watkins. *Programming Flash Communication Server.* O'Reilly Media; First Edition, P. xii, 2005.

[13] A. J. Martin. Iranian web crackdown drives surge in privacy technology. https://news.sky.com/story/ iranian-web-crackdown-drives-surge-in-privacy-technology-11191740, 2019. [Online; As seen on 05 February 2019].

[14] D. Moore and T. Rid. Cryptopolitik and the darknet. *Global Politics and Strategy* Volume 58, 2016 – Issue 1, 2016.

[15] Nmap.org. Nmap manual – chapter 14. understanding and customizing nmap data files. https://nmap.org/book/nmap-services.html, 2019. [Online; As seen on 03 January 2019].

[16] Nmap.org. Nmap manual – chapter 15. nmap reference guide. https://nmap.org/book/man-version-detection.html, 2019. [Online; As seen on 03 January 2019].

[17] "Office Journal of the European Union". Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679, 2016.

[18] G. Owen and N. Savage. Empirical analysis of tor hidden services. *IET Information Security* (Volume: 10, Issue: 3 , 5 2016), 2015.

[19] pcmag.com. Police shut down the wall street market, a top dark web site. https://www.pcmag.com/news/368151/police-shut-down-the-wall-street-market-a-top-dark-web-site, 2019. [Online; As seen on 03 May 2019].

[20] Securedrop.org. Secure drop – share documents securely with these organizations. https://securedrop.org/, 2019. [Online; As seen on 05 February 2019].

[21] Speedguide.net. Port 1111 details. https://www.speedguide.net/port.php?port=1111, 2019. [Online; As seen on 01 February 2019].

[22] ProtACT Team and InTELL Team. Large botnet cause of recent tor network overload. https://blog.fox-it.com/2013/09/05/large-botnet-cause-of-recent-tor-network-overload/, 2013. [Online; As seen on 10 November 2018].

[23] Torproject.org. Metrics torproject.org. https://metrics.torproject.org/, 2018. [Online; As seen on 16 November 2018].

[24] Torproject.org. Tor 0.3.2.9 is released: We have a new stable series! https://blog.torproject.org/tor-0329-released-we-have-new-stable-series, 2018.

[25] Torproject.org. Tor rendezvous protocol, version 2. https://github.com/torproject/torspec/blob/master/rend-spec-v2.txt, 2018. [Online; As seen on 09 November 2018].
[26] Torproject.org. Tor rendezvous protocol, version 3. https://github.com/torproject/torspec/blob/master/rend-spec-v3.txt, 2018. [Online; As seen on 09 November 2018].
[27] Torproject.org. Configuring onion services for tor. https://www.torproject.org/docs/tor-onion-service.html.en, 2019. [Online; As seen on 23 January 2019].
[28] Torproject.org. Tor dev manual. https://www.torproject.org/docs/tor-manual-dev.html.en, 2019. [Online; As seen on 03 January 2019].
[29] Torproject.org. User metrics. https://metrics.torproject.org/userstats-relay-country.html, 2019. [Online; As seen on 01 February 2019].
[30] Wikipedia. Support-vector machine. https://en.wikipedia.org/wiki/Support-vector_machine. [Online; As seen on 28 November 2019].
[31] W. Zamora. Trickbot takes over as top business threat. https://blog.malwarebytes.com/101/2018/11/trickbot-takes-top-business-threat/, 2018. [Online; As seen on 03 February 2019].
[32] zona.media. Roskomnadzor blocked the website “rospravosudie” on complaint about the publication of personal data. https://zona.media/news/2018/07/18/rospravosudie, 2018. [Online; As seen on 05 February 2019].

Biographies

Martin Steinebach is the manager of the Media Security and IT Forensics division at Fraunhofer SIT. From 2003 to 2007 he was the manager of the Media Security in IT division at Fraunhofer IPSI. He studied computer science at the Technical University of Darmstadt and finished his diploma thesis on copyright protection for digital audio in 1999. In 2003 he received his

PhD at the Technical University of Darmstadt for this work on digital audio watermarking. In 2016 he became honorary professor at the TU Darmstadt. He gives lectures on Multimedia Security as well as Civil Security. He is Principle Investigator at ATHENE and represents IT Forensics and AI security. Before he was Principle Investigator at CASED with the topics Multimedia Security and IT Forensics. In 2012 his work on robust image hashing for detection of child pornography reached the second rank "Deutscher IT Sicherheitspreis", an award funded by Host Görtz.

Marcel Schäfer serves as Senior Research Scientist for the Fraunhofer USA Center for Experimental Engineering CESE in Maryland since 2019. From 2009 to 2018 he was with Fraunhofer Institute for Secure Information Technologies SIT in Germany. With a Master's degree in mathematics from the University of Wuppertal, Germany and a PhD in computer science from the Technical University of Darmstadt, Germany, he consults and teaches for topics on dark web, privacy networks and anonymous communication, and also serves as a subject matter expert for privacy, e.g. GDPR and data anonymization. As PI, Co-PI and researcher Dr. Schäfer has lead and worked in various projects that discover new challenges and opportunities broadly spread over the fields of cybersecurity and software engineering in both the public and private sector.

Katharina Brandl studied computer science in Marburg and finished her master degree in 2012. During her studies she was part of the programming languages research group of Prof. Ostermann where she also wrote her master thesis about a type system for parametric tree grammars. Since 2017 she is part of the PANDA project at the Fraunhofer SIT. The PANDA project is an interdisciplinary project researching the darknet and there she is responsible for the computer science part of the project.