On the state of V3 onion services

Tobias Hoeller
Johannes Kepler University Linz
Linz, Austria
tobias.hoeller@ins.jku.at

Michael Roland
Johannes Kepler University Linz
Linz, Austria
michael.roland@ins.jku.at

René Mayrhofer
Johannes Kepler University Linz
Linz, Austria
rene.mayrhofer@ins.jku.at

ABSTRACT

Tor onion services are a challenging research topic because they were designed to reveal as little metadata as possible which makes it difficult to collect information about them. In order to improve and extend privacy protecting technologies, it is important to understand how they are used in real world scenarios. We discuss the difficulties associated with obtaining statistics about V3 onion services and present a way to monitor V3 onion services in the current Tor network that enables us to derive statistically significant information about them without compromising the privacy of individual Tor users. This allows us to estimate the number of currently deployed V3 onion services along with interesting conclusions on how and why onion services are used.

CCS CONCEPTS

• **Networks** → **Network measurement**; Network monitoring; • **Security and privacy** → *Pseudonymity, anonymity and untraceability*; *Privacy-preserving protocols*;

1 INTRODUCTION

Tor onion services enable individuals to operate publicly reachable servers without disclosing their network location. Historically, they have been a sideline of the work done by the Tor project. Some have even claimed that onion services were originally conceived as a demonstration of interesting applications that could be built on top of a free and open network like Tor [2]. This sentiment is also supported by their own statistics which show that in 2021 onion services accounted for only 6 Gbit/s of traffic within the Tor network [9]. This pales in comparison to the almost 300 Gbit/s of bandwidth that the Tor network currently consumes in total.

In stark contrast to these numbers, the public opinion often considers onion services a significant building block of the "Darknet" which is believed to be several times larger in size than the easily accessible parts of the Internet. While it is commonly accepted that this perception is incorrect, it does show that reliable figures on the state of the Tor network and onion services in particular are of interest to a lot of parties.

Unfortunately, the desire to collect this information directly conflicts with the fact that onion services are designed to avoid data collection as much as possible so there is actually a very limited amount of information about onion services that is gathered and published by the Tor project. Currently, the only collected metrics are the number of V2 onion services which were around 200,000 in the first months of 2021 and the amount of traffic generated by V2 and V3 onion services [9].

In the past there have been several other research efforts to learn more about how onion services are being used [6, 7], but they all focused on V2 onion services. This is mainly caused by the fact that certain weaknesses in V2 onion services made it easier to collect and analyze data about them. Since there are no similar issues known about V3 onion services, we know much less about the current version of onion services than we knew about the previous version.

A simple and obvious example would be the total number of active onion services in the Tor network. Right now, we have a solid estimate on the number of V2 onion services but have no information about V3. This is especially relevant, because V2 onion services will be discontinued in 2021 [3] leaving the research community with no information on how many onion services are currently running.

This work tackles the challenge of collecting basic information about V3 onion service usage like the number of currently running V3 onion services and the amount of users they have.

We first discuss the improvements introduced by V3 onion services that make gathering and interpreting data about onion services harder. In section 3 we describe our measurement setup in detail. Afterwards, we present a detailed analysis of our collected data which answers several open questions about V3 onion services.

2 TOR AND ONION SERVICES

Tor is an onion routing technology that anonymizes network traffic by tunneling it via several nodes. A connection established via the Tor network is referred to as *circuit* and usually consists of three nodes. The currently available members of the Tor network are defined by the *consensus*, a document that is created by a selected small group of trusted relay operators called *directory authorities*. This consensus is published every hour and lists all currently known relays along with all the information needed to create circuits through them. Additionally, the consensus assigns *flags* on relays based on their behavior and capabilities. The most important flags in the context of this paper are *Fast*, *Stable*, and *HSDir*. A relay is considered fast if it has a bandwidth of more than 105 KB/s, stable if it has a weighted mean time between failure of more than 7 days, and HSDir if it is stable, fast, and has an uptime of more than 96 hours. Of special importance when talking about onion services is the fact that the consensus also includes a shared random value which changes every 24 hours to ensure that certain parts of the Tor network remain unpredictable.

Onion services operate on top of the Tor network by bouncing both incoming and outgoing connections via other Tor relays to hide their network location. In order to accept incoming connection

FOCI'21, August 27, 2021, Virtual Event,USA

ACM ISBN 978-1-4503-8640-1/21/08.
https://doi.org/10.1145/3473604.3474565

requests indirectly, every onion service selects random relays as *introduction points* and establishes circuits to them. These circuits must remain intact as long as the onion service keeps running since the only way to contact an onion service is via its introduction points.

Since introduction points must be replaced if they go offline, Tor needs a way to automatically discover the current introduction points for a specific onion service. The identifier of the onion service along with metadata and the introduction points is packaged in a *service descriptor* by the onion service and published. To enable publication, a set of Tor relays which have been granted the HSDir flag in the consensus form the hidden service directory, a distributed hash table, where service descriptors can be posted and retrieved. By default, V3 service descriptors are uploaded to four consecutive relays at two different positions (replicas) within the distributed hash table. In order to connect to an onion service, a Tor client needs to know the identifier of the service descriptor which is more commonly known as the onion address. With that address the client can request service descriptors from the hidden service directory and contact the onion service. To ensure that the chosen introduction points cannot monitor incoming connections on a hidden service directory, clients have to select a random Tor relay as *rendezvous point* and ask the onion service via one of its introduction points to establish a connection through the rendezvous point. Both the client and the onion service create a circuit to the rendezvous point which is only responsible for connecting the two circuits to establish a secure and anonymous connection between the onion service and a client.

2.1 Flaws in V2

In the previous version of onion services, service descriptors were identified by the .onion address of the service they belonged to. This enabled any Tor relay that was a member of the hidden service directory to extract currently valid onion addresses from the uploaded descriptors. Since there is no other way to learn about available onion addresses, other than being told by the owner of the onion service, this leak of information is critical because it enables both attackers and researchers to enumerate all currently existing V2 onion services.

Tightly coupled with the first problem is the fact that V2 service descriptors are not encrypted. So every relay on the hidden service directory that receives the descriptor can use it to directly connect to the onion service even without knowing the address (which is irrelevant for V2 onion services since V2 service descriptors are identified by the onion address).

These two information leaks enabled the mass collection of onion addresses for any parties with the capability of operating Tor relays with the HSDir flag. Consequently, running a relay only to collect and probe onion addresses is considered malicious[1] behavior by the Tor project and such relays are actively removed from the consensus. These weaknesses were also used by researchers to collect the onion addresses of active V2 onion services [1, 7] and establish connections to them in order to learn about the services they provide.

[1]https://community.torproject.org/relay/community-resources/bad-relays/

Another aspect of the hidden service directory that made V2 onion services vulnerable, was the fact that it assigned the responsibilities for descriptor space based on the relay's fingerprint. Since the fingerprint is chosen by the relay, malicious actors could modify the position of their relays within the hidden service directory by carefully choosing their fingerprints. This issue was best highlighted by the shadowing attack performed by Biryukov et al. [1], who operated dozens of relays in parallel. Usually this would not be a problem because at most two relays in a consensus can have the same IP address so operating more relays does not impact the consensus at all. But those inactive relays could still collect flags while they were not part of the consensus and by temporarily disabling individual relays Biryukov et al. could control which two relays would join the hidden service directory at any given time. This enabled them to collect 100% of all V2 .onion addresses within only 24 hours because they had full control over which part of the HSDir they were monitoring at any given time.

Finally, the way V2 service descriptors are distributed across the hidden service directory turned out to be problematic. In order to make sure that the responsible HSDir relays for a service descriptor change regularly, the position of a V2 descriptor within the hidden service directory is derived from the identifier of the onion service and the current date. While this works well to move the descriptor around, this approach has the massive problem of being predictable. An attacker can easily calculate the future positions of a specific onion service's service descriptor. This was doubly problematic with the previously mentioned shadowing attack because it enabled a malicious actor to constantly become the responsible hidden service directory relay for a specific onion service. Even without the shadowing attack, this is possible for an attacker with access to enough public IP addresses.

2.2 Changes in V3

Considering all the issues with the hidden service directory that came up in V2 onion services, it comes as no surprise that significant changes were made to make V3 onion services more resilient against malicious HSDir relays.

To prevent members of the hidden service directory from collecting onion addresses, a key derivation scheme was introduced. Instead of using the onion address, which is just an encoded public key, as identifier for the service descriptor, a new *blinded public key* (*bpk*) is derived from the current shared random value in the consensus and the V3 onion address. Clients with knowledge of the onion address can easily generate the blinded public key themselves and request it from the hidden service directory while the relays within the HSDir are unable to derive the original key from the uploaded blinded public key. Additionally, this change makes it impossible for an attacker to precompute future descriptor locations for a specific onion service since those depend on the identifier of the descriptor, which is no longer predictable thanks to the shared random value.

Using two different keys also provides a possibility to prevent HSDir relays from reading the contents of service descriptors. There is a private key counterpart to every public key that is encoded in an onion address. This private key is only known to the onion service and can be used to encrypt service descriptors before uploading

them. This is no issue for legitimate clients since they need to know the address anyway to request the descriptor. However, malicious HSDir relays can no longer probe onion services, since they have no way to decrypt the introduction points within the descriptor.

To ensure that relay operators have no influence over their positions within the hidden service directory at any given time, descriptor space responsibilities within the HSDir are now also impacted by the shared random value. Since a relay needs to run for at least 96 hours before joining the hidden service directory it is impossible to generate a new relay that joins the hidden service directory before the network moves on to a new shared random value.

3 EXPERIMENT SETUP

From the changes highlighted in the previous section it becomes quite clear that most attempts to obtain information about V2 onion services have focused on the hidden service directory and the V3 onion service specification has gone to great lengths to reduce the amount of information that can be obtained from monitoring the hidden service directory. Nevertheless, as long as every onion service has to publish their descriptor inside the HSDir and every client must fetch them from there as well, the HSDir will leak some information to the participating relays.

3.1 Technical Details

For our experiment, we deployed a set of 50 Tor relays (family: `008196DC449482C73CFA9712445223917F760921`) which meet the requirements to obtain the HSDir flag and log every upload and download of a V3 service descriptor. Instead of writing those logs to disk, they are directly handled by a log listener attached via the Stem[2] library which extracts relevant information from the descriptors, sorts them alphabetically and stores it in a SQL database. Just like the relays, this database was operated by us within our own network on our own hardware. For every upload we store the blinded public key, the relay that received the upload, and a timestamp that only contains the year, month, day and hour of the upload. For downloads we store the same data, except that the timestamp are reduced to daily granularity.

3.2 Privacy considerations

When we designed our experiment, we had to deal with the risk of endangering the privacy of Tor users through our data collection. Thankfully, the Tor project has their own advisory board[3] which helps researchers to conduct their experiments in a safe way. They reviewed our experiment setup and suggested several privacy improvements. We continued to implement these changes until the advisory board no longer had any objections. In this section, we discuss the most important questions we encountered and how we decided to address them.

3.2.1 Data Management. Access to the machines involved in the experiment was exclusively granted to the members of our research team responsible for the experiment. The database with the collected information will be retained as long as our research is ongoing. Once the research is concluded, the raw data will be erased permanently.

3.2.2 Traffic correlation. On a basic level, our stored data reveals which requests were made to our relays at which time. This might allow an attacker to combine this information with other data sources to launch time-based traffic correlation attacks. To mitigate this issue, we decided to drop all metadata about the requests and to truncate timestamps at the hour value for all uploads. For downloads we saw a greater risk of attack, because they have to be triggered by clients manually, while uploads happen automatically and semi-regularly. Therefore, we decided to cut upload timestamps at the day value instead of the hour. This should render our data useless for time-based correlation attacks, without impacting statistical significance.

Another potential source of correlation raised by the Tor research safety board is the order in which blinded public keys are entered into the database. If that order was the same in which the data was received, this would give attackers an alternative way to accurately determine the time certain requests were made. This is mitigated by sorting blinded public keys alphabetically, before inserting them into the database.

3.3 Hardly used onion services

Some onion services are used for very specific tasks that require only a single user making occasional connections. If one of our relays is a responsible HSDir for such a service and the client selects our relay to request the service descriptor from, our data reveals when an onion service was used. Paired with knowledge about the purpose of an onion service belonging to a single user, this alone might reveal more information about the user than we intended. Unfortunately, we have no way of knowing this in advance, so we cannot exclude such cases during our raw data collection. Consequently, we limit access to our raw data to the researchers responsible for the experiment and make sure that in publications only aggregated information on barely used onion services is published.

3.4 Unwanted attention

While our collected information does not contain any onion addresses, attackers with knowledge of onion addresses could easily link our collected blinded keys to addresses they know about. This allows them to use our data to estimate the popularity of an onion service. Since one of the goals of onion services, is keeping the number of users private [4], making this information publicly available could violate the privacy of some onion service operators. Ultimately, this violation could lead some attackers to specifically target onion services because of their popularity within our data.

While this could have been avoided entirely by not storing blinded public keys, we decided to include them in our raw data to enable research on the usage of well known onion addresses and their development over time. This development is especially interesting in the current transition from V2 to V3 onion services. To prevent abuse of our data we will never publish blinded public keys directly (since we have no way of knowing if anyone else will link them to an onion address) and only publish information on

[2] https://stem.torproject.org/
[3] https://research.torproject.org/safetyboard/

how many users an onion service has, if that does not constitute a risk to that service.

3.5 Data publication

From the previous sections is becomes clear that publication of the unfiltered raw data is not desirable, because some entities might be able to instrument our data to harm individuals. Nevertheless, we would still like to make as much of our data as possible available to other researchers, so we are currently working on creating a processed data set from our raw data that enables statistical analysis without endangering the privacy of individual Tor users. This would allow us to keep data from the experiment after the raw data has been deleted.

4 RESULTS

While our relays are still running and collecting more data, the already collected data is sufficient to answer several questions surrounding V3 onion services.

4.1 Uploads

The first metric we derive from our data is an estimate of the number of running V3 onion services. Before discussing it in more detail, it should be noted that since version 0.4.6.1-alpha[4] Tor does also collect metrics about the number of V3 onion service, but so far no statistics have been published. In the future, we expect the Tor Metrics [9] team to provide a more reliable estimate on the number of V3 onion services, which can be used to cross-check our own results. To make this comparison as easy as possible, we adapted the specification on how Tor Metrics generates statistics on V2 onion services [5] and only made the necessary changes to be compatible with V3.

First, we need to know the fractions of descriptors seen by our members of the hidden service directory based on the hash value ($hsdir\_hash_x$) that determines their position within the HSDir. Like regular Tor clients we can derive the HSDir from a Tor consensus [8] and calculate the fractions of our own relays as

$$hsdir\_share_x = (hsdir\_hash_x - hsdir\_hash_{x-4})/2^{256}/4.$$

Dividing *hsdir_share* by 4 is necessary to counter the effect that every service descriptor is uploaded to four consecutive relays. To estimate the total size of the hidden service directory, we need to combine our relative share with the number of uploaded blinded public keys (bpk_count_x):

$$extrapolated\_size_x = (bpk\_count_x \cdot (1/hsdir\_share_x))/4.$$

The division of the *extrapolated_size* by 4 is necessary to obtain the number of distinct V3 onion addresses since every service maintains two replicas in two time periods. To aggregate this set of extrapolated sizes to a single value, we opted to use the same weighted interquartile mean [5] that the Tor metrics team uses to estimate the V2 onion address count.

Figure 1 shows the results of our calculations which indicate that the number of V3 onion services was between 600,000 and 700,000 in March and April of 2021 making them about three times more popular than V2 onion services. The drop to zero between

[4]https://blog.torproject.org/node/2011

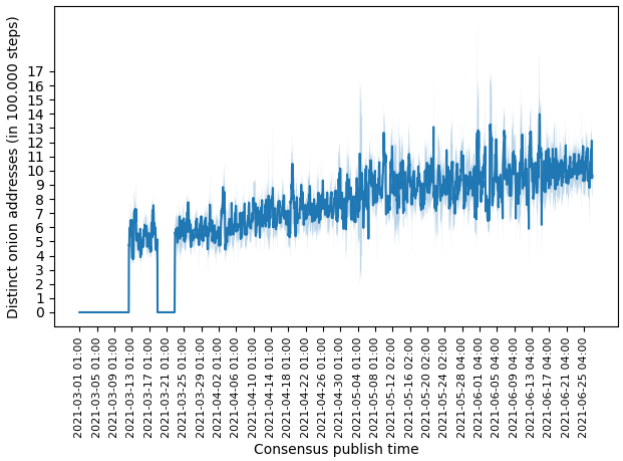

Figure 1: Extrapolated number of V3 onion services

March 18th and March 23rd was caused by a temporary outage of the directory authority *Faravahar*, which caused all of our relays to lose their HSDir flag until the authority came up again. An important criterion to judge the significance of our measurements is the relative share of the hidden service directory we are extrapolating from. Tor metrics define a threshold of 1% for V2 onion services and excludes data for days with less information. Figure 2 shows the relative share of the hidden service directory we monitored during our experiment. The constant decline in share between April and June was caused by an error that lead some of our relays to stop reporting statistics. Once the problem was detected and corrected, our share stabilized at around 0.8% again. Initially, we expected this to cause very unreliable estimates on the size of the hidden service directory for the affected months (especially May), but our data does not support this assumption. When all 50 relays started reporting statistics again, the increased share in the hidden service directory did not have a significant impact on the estimated number of onion addresses. This leads us to conclude that it is possible to obtain acceptably stable estimates on the number of onion services while owning significantly less than 1% of the hidden service directory.

The next interesting piece of information to know about V3 onion services is their average lifetime but, in contrast to previous studies on V2, we have no way of finding out if two blinded public keys belong to the same onion address. However, we do know that similar research on V2 onion services [7] found that most onion services did not live long enough to show up in their data multiple times. While we have no way of confirming these results for V3, we do know that every blinded public key is valid for 48 hours and is re-uploaded at least every 60-120 minutes [8]. Based on this we expect to see every blinded public key uploaded on average 32 times if the Tor network remains stable. Relays joining or leaving the HSDir can cause us to see fewer uploads and unstable introduction points increase the number of observed uploads. Figure 3 shows that during our experiment the amount of descriptors seen between 3 and 35 times was fairly constant, which indicates that these were caused by the dynamics of the hidden service directory. The spike at 38 and 39 uploads per bpk seems to indicate that this is the average

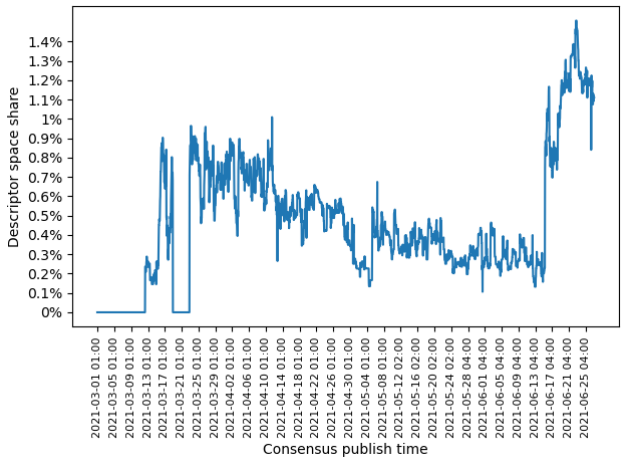

Figure 2: Relative share of HSDir we observed

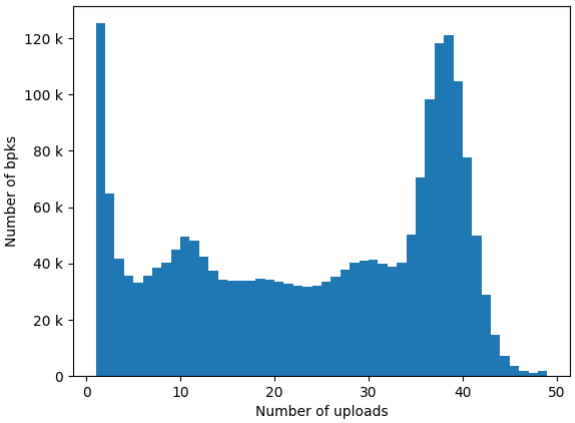

Figure 3: How often are bpks uploaded

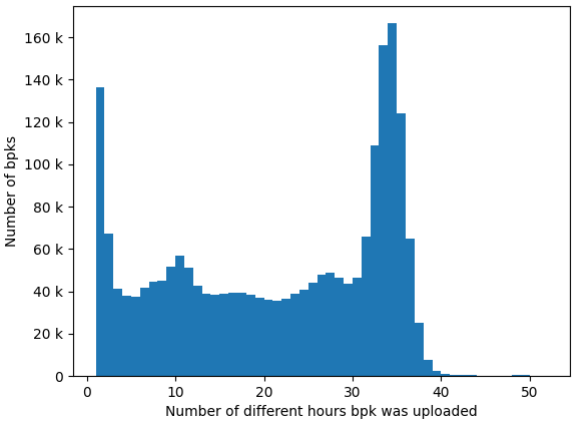

Figure 4: For how many different consensuses are bpks uploaded

number of uploads for a stable onion service. The high amount of bpks with a single upload supports the theory that there are a lot of onion services that live for a very short time. We speculate that these instances are either created by people experimenting with onion services or that there are onion services which are only used once and thus have no need to republish their descriptor.

It should be mentioned that we did detect a small number of blinded public keys that were uploaded more than 50 times to our relays with the record holding key being re-published 16951 times within 48 hours. But since more than 99.5% of all bpks were uploaded between 1 and 50 times, we feel confident that we can ignore larger numbers as symptoms of misconfiguration of the responsible Tor client.

This caused us to wonder if there are V3 onion services that publish the same bpk over more than 48 hours. Figure 4 shows in how many different hours (consensus freshness periods) our observed bpks. In this case we did not have to exclude outliers for plotting since no blinded public key was published more than 53 times.

Comparing Figure 3 and Figure 4 we can see that we get much closer to our expected 32 uploads on average if we ignore re-uploads in the same consensus freshness period that were most likely caused by changing introduction points or issues with the Tor client of the onion service. The delta between Figure 3 and Figure 4 can therefore be interpreted as a rough estimate for how often onion services have to re-publish their onion service for other reasons.

4.2 Downloads

In order to assess the relevance of onion services, it is essential to estimate the amount of users they are handling. There are no official statistics on how many users onion services have collected by the Tor network and previous research [6, 7] has focused on what onion services are used for, so there is little data on the usage of onion services.

We can provide some insight into how frequently onion services are being accessed by investigating the number of times every blinded public key was requested from our nodes. It should be noted that descriptor downloads do not correspond to visits since the Tor client caches descriptors, but also not to visitors since there are several reasons that could cause Tor to request a descriptor multiple times. Even more specific, the download of a descriptor only tells us that a Tor client intended to connect to an onion service, not if it actually connected. So we interpret the count of descriptor downloads as an upper bound on the number of visitors and a lower bound on the number of visitation attempts.

Tor clients use only 6 out of the 8 responsible HSDir relays for downloading (the other two are there in case a relay goes offline). Usually, one of those 6 relays is chosen at random, so our recorded number of descriptor downloads must be multiplied by a factor of six to extrapolate the actual number of descriptor requests for a specific blinded public key. The only exception to this rule are requests for onion services that no longer exist. In this case a Tor client will try all six responsible hidden service directories before giving up, so there is no need to extrapolate the number of download attempts.

Table 1: Received upload and download requests (in absolute and relative numbers)

Description	Requests		BPKs	
Total Uploads	51,415,871	100%	2,041,638	100%
Used Uploads	9,731,936	19%	294,245	14%
Unused Uploads	41,683,935	81%	1,747,393	86%
Total Downloads	95,411,136	100%	1,310,062	100%
Successful Downloads	29,326,445	31%	293,903	22%
Failed Downloads	66,084,691	69%	1,016,159	78%

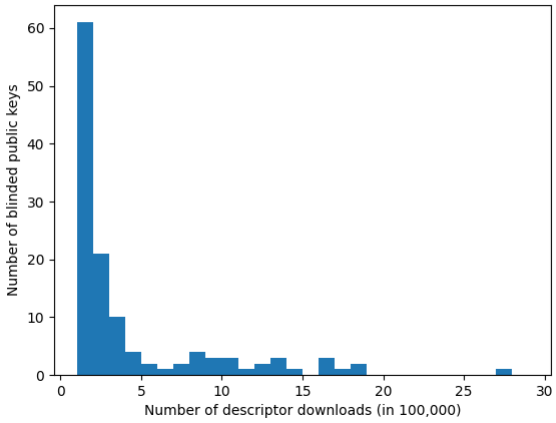

Figure 5: Top 0.01% of most downloaded bpks

Since we already learned from the uploads that onion services are very dynamic we expected a high number of onion services with a very low number of downloads. Table 1 confirms this theory since 86% of all blinded public keys that were uploaded during our experiment were never downloaded by a client. When interpreting this result, it is important to remember that we usually only control one out of eight responsible hidden service directory nodes, so an unused upload does not mean that the service was never downloaded, it just means it was never downloaded from our node. V3 descriptors are always uploaded to 8 different nodes, but only downloaded from 6 (the other two serve as backup if a HSDir node goes offline), so about 25% of our uploads are meant to be unused. The remaining unused uploads are most like caused by onion services that are either unused or used so rarely that our node was never chosen by chance and remained unused.

Next, we took a look at the most downloaded blinded public keys to quantify how successful the most popular onion services are. We found that more than 77.5% of all download requests received by our relays were asking for the most popular 1% of blinded public keys with the record key being requested more than 1.6 million times within 48 hours. This implies that at least 9.6 million attempts to visit the service were made during one day. Figure 5 shows the top 0.01% of most downloaded blinded public keys which make up 47% of all downloads for blinded public keys. This illustrates nicely that a very small amount of onion services is responsible for most onion service usage. While we have no way of knowing what kind of onion service would be so popular, it seems fair to speculate some of them might be command and control servers that are used by botnets. Previous researchers have already found that V2 onion services were used to control botnets [7] and it would be logical that V3 onion services continue to be used in this way. A possible way to support this theory, is to quantify the number of requests for blinded public keys that were never published. In contrast to humans, programs tend to repeatedly try to connect to a no longer working onion service while humans will give up after a short time. In our data set, there are several blinded public keys that were never published to our relays, but still requested more than one million times over 48 hours from just one of eight responsible hidden service directories. We see two possible explanations for this observation: Either these keys belong to a defunct botnet server or they were victim of a DoS attack that knocked out the onion service in the previous time period but did not stop trying to attack.

Table 1 also shows that more than two thirds of all received download requests could not be answered because the requested descriptor had not been uploaded. Since we do not track the hours of download requests, we can only confirm failed download requests if the descriptor was never published, so our results only put a lower bound on the total number of failed download requests. While it is possible that descriptors for running onion services are requested from our relay without ever being uploaded to it, this requires very unfortunate timing of relays joining and leaving the hidden service directory. The large share of failed requests indicates that a significant share of requested onion addresses belongs to disabled/nonexistant onion services. This observation also applies to the blinded public keys depicted in Figure 5 indicating that a majority of onion service requests is asking for a small number of onion services that mostly do not exist.

The fact that so many onion service requests fail paired with the justified assumption that many high-volume onion services are only used for machine-to-machine communication means that we are unable to derive at any conclusions about how many human users connect to onion services.

5 CONCLUSION

Our results show that despite the additional steps taken by version 3 of the onion service protocol, it is still possible to collect statistically significant information about Tor onion services and their users by monitoring the hidden service directory. We present an experiment design that enables us to collect this information with reasonable effort and without compromising the privacy of individual Tor users. We have provided a first estimate on the current number of deployed V3 onion services and reveal that many of them only exist for short periods of time. By evaluating the download requests received by the hidden service directory, we demonstrated that most service descriptors are hardly ever downloaded, most likely because they are used by single users or small groups. On the other hand we confirmed that a very small amount of onion services is responsible for most onion service activity. We believe that future research should try to identify the cause of those high-volume onion services in order to deepen our understanding of how onion services are being used.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers at FOCI'21 for their high-quality, valuable feedback. In particular, we thank our shepherd, Roger Dingledine, for his guidance on improving the paper.

This work has been carried out within the scope of Digidow, the Christian Doppler Laboratory for Private Digital Authentication in the Physical World, funded by the Christian Doppler Forschungsgesellschaft, 3 Banken IT GmbH, Kepler Universitätsklinikum GmbH, NXP Semiconductors Austria GmbH, and Österreichische Staatsdruckerei GmbH.

REFERENCES

[1] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. 2013. Trawling for tor hidden services: Detection, measurement, deanonymization. In *2013 IEEE Symposium on Security and Privacy*. IEEE, Berkeley, CA, USA, 80–94. https://doi.org/10.1109/SP.2013.15

[2] Roger Dingledine. 2017. Next Generation Tor Onion Services. Presentation at DEF CON 25. (2017). https://media.defcon.org/DEFCON25/DEFCON25presentations/DEFCON25-Roger-Dingledine-Next-Generation-Tor-Onion-Services-UPDATED.pdf

[3] David Goulet. 2020. Onion Service version 2 deprecation timeline. The Tor Project Blog. (2020). https://blog.torproject.org/v2-deprecation-timeline

[4] David Goulet, Aaron Johnson, George Kadianakis, and Karsten Loesing. 2015. *Hidden-service statistics reported by relays*. Tor Tech Report 2015-04-001. The Tor Project. https://research.torproject.org/techreports/hidden-service-stats-2015-04-28.pdf

[5] George Kadianakis and Karsten Loesing. 2015. *Extrapolating network totals from hidden-service statistics*. Tor Tech Report 2015-01-001. The Tor Project. https://research.torproject.org/techreports/extrapolating-hidserv-stats-2015-01-31.pdf

[6] Sarah Jamie Lewis. 2016. OnionScan: Investigating the Dark Web. (2016). https://onionscan.org/

[7] Gareth Owen and Nick Savage. 2016. Empirical analysis of Tor Hidden Services. *IET Information Security* 10, 3 (2016), 113–118. https://doi.org/10.1049/iet-ifs.2015.0121

[8] The Tor Project. 2017. Tor Rendezvous Specification - Version 3. (2017). https://github.com/torproject/torspec/blob/master/rend-spec-v3.txt

[9] The Tor Project. 2021. Tor Metrics. (2021). https://metrics.torproject.org