Tor Relay Analysis and Guard Prediction

Cassian Corey
University of Massachusetts
Amherst
cjcorey@umass.edu

Divyesh Harit
University of Massachusetts
Amherst
dharit@umass.edu

ABSTRACT
In this paper, we analyze properties of currently running Tor relays. We then attempt to expose new details about the process used by Tor directory authorities to assign Guard flags to relays. By applying machine learning classification, we find that Guard flags can be predicted with high accuracy using other relay properties. The use of decision trees for this classification allows us to easily understand the logic used. We argue that the process of constructing a classifier and observing its logic can be repeated on-demand and with ease by an attacker thereby allowing them obtain up-to-date information regarding what properties they should advertise to boost their chances of being assigned the Guard flag.

CCS Concepts
• Networks->Network privacy and anonymity. • Networks->Network performance evaluation->Network measurement.

Keywords
Tor, privacy, bandwidth, lifespan

1. INTRODUCTION
1.1 Censorship and Tor Background
Censorship exists for a variety of reasons, among which politics and religion are the most common. Various types of network censors exist to prohibit users within their control from accessing certain websites and data over the Internet. Not only can users not access these websites, if they are caught trying to do so they may face a lifetime of imprisonment or other serious consequences [1].

Despite this threat, users still attempt to circumvent censorship through a variety of means, one of which is Tor (also known as The Onion Router). Tor works by wrapping data in encrypted layers, like the layers of an onion. This structure protects the contents of the data while also preserving sender anonymity. At each node in the encrypted packet's path through the Tor network, its layers are peeled off one by one. To prevent one node from learning too much information, there are always at least three nodes in any path through the Tor network: a guard node, a middle node, and an exit. These nodes are also known as relays and can be run by anyone in the world with access the necessary tools.

Tor operates entirely on volunteer bandwidth. Each volunteer advertises their relay with certain properties. These properties are then voted upon by a group of authorities called Tor directory authorities. Once every hour, the Tor directory authorities publish the extent of their knowledge and measures of agreement on the current state of the Tor network in a document called the Tor consensus file. This file is what Tor users consult, along with the Tor Path Specification protocol, when deciding which relays to include in their path [2].

As a result, relays that want to receive high amounts of traffic care deeply about the information that gets published about them in the consensus file. To establish a reputation and see more traffic, they must find a way to ensure that this information is accurate, up-to-date, and favorable.

Volunteers who host relays often have a goal in mind for which position they would like to occupy in a user's path to the Internet. The safer positions are that of entry and/or middle relay. These relays have no knowledge of or ties to the final destination of the packets they forward. They only know the next hop in the path they have been chosen for. Bold volunteers willing to establish connections from their machine to the multitude of Internet destinations being accessed by Tor users will host exit relays.

Whether hosting an exit or a middle relay, the end goal is typically to become a guard relay. Guard relays serve as entry points into the Tor network and were introduced in response to a weakness discovered in the path-creation method used by Tor. This weakness left Tor users more susceptible to encountering a malicious relay who may deanonymize them using traffic correlation attacks. In the original method for path construction, users built entirely new paths each time they wished to send their connection through Tor. Although the probability of encountering a malicious relay is slim given the size of the network, as the number of paths being used increases, so does the likelihood of encountering a malicious relay. Details of this weakness are described in [2].

Because of their need to be trusted and reliable, becoming a guard node is challenging and requires dedication from the relay host. It often takes more than 68 days before a new relay gets the guard flag. In fact, a minimum number of days is required before the flag will even be considered. In 2013, the number was approximately eight days [3]. Benign and malicious relay hosts are both constantly vying to earn a Guard flag. For benign hosts, it represents a sense of accomplishment. For malicious hosts, it is an opportunity to carry out several varieties of deanonymization attacks [4].

1.2 Motivation
As far as we are aware, there has not been much research in studying changes in trends of relay properties and thus observing evolution of Tor network over the past few years. Analyzing relay properties allows us to observe any existing trends or correlations and draw appropriate conclusions about the state of the network. Due to its role in evading censorship, it is important that Tor be observed frequently for trends and anomalies. Doing so would allow for constant improvement of the Tor network, thus being one step ahead of the censors. Additionally, a current analysis of Tor relays – which make up the majority of the Tor network [5] – may prove useful to researchers trying to improve or create Tor simulations like those used in Shadow [6].

A critical step in machine learning is the collection and preprocessing of data. In addition to keeping users updated, analyzing relay properties allows us to better extract which properties to use as features in our guard flag classifier.

Being able to successfully predict guard status using relay properties that are publicly available for any user to see would help in increased security against malicious attackers, as guards play a crucial role in protecting the anonymity of the user. If acquiring the guard flag can be predicted to some accuracy, known attackers or suspicious users can be denied that flag, thus evading their possible deanonymization attacks.

1.3 Sections

In Section 2 we present related work analyzing Tor relays and using classification to predict Guard flags as a means of attack. Section 3 describes our methodology in data collection, selection of features from this data, and finally model selection based on the findings in our analysis. Section 4 presents the results of our relay property analysis as well as results from our machine learning experiments. In section 5 we give some concluding remarks, describe missed opportunities, and propose future directions this project could take.

2. Related Work

2.1.1 Tor Relay Analysis

There has been significant research done in the area of Tor, and some on its relays and their selection algorithms. Wacek et al. [7] created their own scaled version of Tor and evaluated various relay selection algorithms. Anupam Das, Nikita Borisov et al. [8] proposed a reputation based model called Re^3 that on the basis of past-user experiences, can be used by Tor to filter out less reliable relays. Elahi et al. [9] studied the entry guard relay selection and its robustness against attacks. Exit relay scanners were developed by Winter et al. [10] to show that there are multiple malicious exit nodes operating in the Tor network.

2.1.2 Classification as an Attack Mechanism

To our knowledge, there is no attack on Tor that analyzes the inner-logic of machine-learning models to reverse-engineer the specific relay property measurements used by Tor directory authorities to assign Guard flags.

3. METHODOLOGY

3.1 Data Collection

All data was collected using the Onionoo Tor network status protocol. This protocol provides information about currently running Tor relays and bridges in the form of documents. We construct our sample set of relays and their properties from the Onionoo details document. More information regarding how these requests can be made is available on the Onionoo protocol website [Onionoo]. Collected properties with a brief description are listed in Table 1. Certain descriptions are borrowed directly from the protocol website.

Table 1. Gathered Relay Properties

Fingerprint	Onionoo's unique relay identifier
Nickname	The canonical nickname if present
Age	Number of seconds since relay was first seen in Consensus
Uptime	Time in seconds since relay was last restarted
Address/Port Uptime	Time in seconds since relay last changed its port or IP address
Advertised Bandwidth	Minimum of observed bandwidth, bandwidth burst, and bandwidth rate reported in B/s and used to calculate consensus weight [Path Spec]
Consensus Weight Fraction	Fraction of consensus weight given to this relay (0 to 1)
Ports Accepted	If it is an Exit relay, this is the number of ports through which it allows connections to the Internet (0 to 65535)
AS Number	The autonomous system this relay belongs to
Effective Family	Number of relays that are in a mutual family relationship with this relay

Presence of the following flags in each relay's details was also recorded: BadExit, Exit, Fast, Guard, HSDir, Stable, Running, Valid, and V2Dir. Our final relay sample set contained a total of 8382 relays. This number includes all Tor relays known about in the consensus file and currently running at the time of collection.

Our goal in analyzing relay properties was to find best-fit distributions for each relay and any correlations between relays. We used Python's Scipy library [cite] to calculate measures of determination and correlation for determining goodness of fit. For individual properties, measures of determination show how accurately the data is represented by a normal distribution. Measures of correlation between pairs of properties show whether or not a linear relationship exists between those two properties.

Some of the properties we collected were better suited to a log-space analysis. When this was the case, we used built-in methods to convert the values to log-space.

The following sections describe how and why we extracted each group of properties.

3.1.1 Flags

Currently, there are 11 flags that can be assigned to a relay by directory authorities. We consider the following nine flags: BadExit, Exit, Fast, Guard, HSDir, Stable, Running, Valid and V2Dir. Descriptions of these flags and their requirements for assignment can be found in [Tor Spec].

3.1.2 Timing Properties

We calculated three timing-related properties: relay age, relay uptime, and relay address/port uptime. Relay age is calculated as the elapsed seconds between when a relay was first- and last-seen in the Tor consensus file. Relay uptime is calculated as the difference between when a relay's information was requested and when that relay was last restarted. Similarly, address/port uptime is the elapsed seconds between when the relay information was requested and when that relay last changed its port or IP address.

3.1.3 Reported Bandwidths

There are four bandwidth-related relay properties provided by Onionoo's protocol. Bandwidth rate and burst are the average bandwidths a relay is willing to sustain over long and short intervals respectively. The third property, observed bandwidth, reports an actual measurement of the bandwidth a relay can sustain over a ten second period. The smallest of these three is

chosen to be the advertised bandwidth and is the value used when calculating consensus weight.

In addition to analyzing current bandwidth rates for currently running relays, we also pull historical data from Onionoo to observe trends in bandwidth over time.

3.1.4 Consensus Weight

Clients use the consensus weight when deciding which relays to include in their paths. It comes from a secret blend of bandwidth and other properties and is agreed upon by the directory authorities. Since consensus weight units are arbitrary, we extract each relay's consensus weight as the fraction of weight that relay contributes to the overall network.

3.1.5 Ports Accepted

Relays given the Exit tag must also advertise an exit policy specifying the ports through which they allow external connections to the Internet. Onionoo conveniently parses these complicated summaries into a dictionary containing all of the ports a relay accepts, or all of the ports a relay rejects. Instead of considering the actual port numbers being accepted or rejected, we calculate the total number of ports a relay allows connections on. We make the decision to focus on number of ports being accepted or rejected rather than the specific port numbers themselves.

3.1.6 AS and Family Size

The AS that a relay belongs to and it's reported family are crucial to path selection because Tor protocol clearly states that no two relays from the same family shall be used in the same path [cite].

We calculated family size by measuring the list of relays reported as being in the effective or indirect family for a relay.

3.2 Guard Prediction

3.2.1 Feature Extraction

The first step in our machine learning process is to decide which relay properties to use as features when training our model. To avoid bias, we ignore knowledge of Tor's flag-assignment protocol and previous work describing what is necessary to achieve guard status. We do not make any assumptions about which properties might affect whether or not a relay is marked as a guard. Instead we allow for the possibility that any combination of properties may be used to accurately predict Guard status.

3.2.2 Model Selection, Training, and Testing

We apply several machine learning classifiers to our collected relay sample set using Python's SciKit-Learn library [3]. For each classifier, the features are the labels and the following four classifiers were chosen: Naïve Bayes, K-Nearest Neighbors, Logistic Regression, and Decision Trees.

Naïve Bayes was chosen because it assumes that features are independent and their values are distributed according to a Bernoulli process. Naïve Bayes requires that the data be normalized to overcome issues of scale between values.

The second classifier, K-Nearest Neighbors, was chosen because of the potential that certain clusters may exist among relay characteristics. In particular, we assume that all relays in the same relay family report the same list of family members and therefore have the same family-size feature in our relay data. Like Naïve Bayes, K-Nearest Neighbors also requires feature values to be scaled or normalized prior to training.

Decision trees were chosen to be our final model because SciKit Learn makes it easy to extract their underlying logic and this logic is structured in a way that is easy for humans to understand. Decision trees do assume any linear relationship between features and require no normalization or scaling prior to training.

After removing entries with missing features or invalid inputs, we split our data into training and testing sets of equal size and roughly equal ratios of guard to no-guard flags.

4. RESULTS

4.1 Relay Analysis

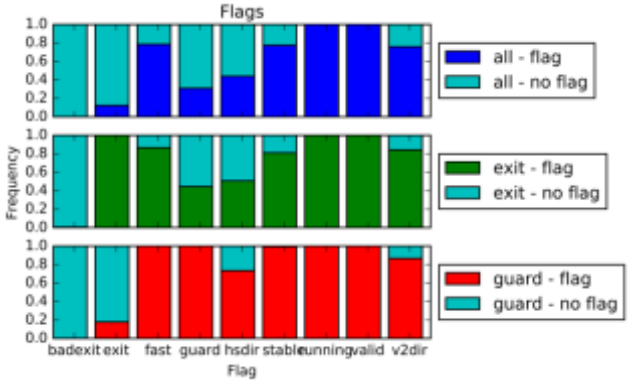
In this section we describe our findings from exploring relay properties and the potential relationships between them. In some cases, distinction is made between type of relay based on whether or not they possess a certain flag (e.g. Guard or Exit flags). When distinguishing between guard and exit relays, and unless otherwise stated, it is possible for a relay in the guard set to also have an exit flag or vice versa.

4.1.1 Flags

The distribution of all flags collected in our dataset is shown in Figure 1 with distinction made between Exit and Guard relays. Our dataset contained 573 exit-only relays (with no guard flag), 2123 guard-only relays (with no exit flag), 458 relays with both exit and guard flags, and 5228 with neither exit nor guard flags. This shows that approximately 12% of all running nodes are relays with exit flags and approximately 31% are relays with guard flags.

The results also show that very few guard relays are also exit relays (18%) but almost half of exit relays are also guard relays (44%).

Figure 1. Presence of flags by relay type.



4.1.2 Timing Properties

The distributions of all relay timing properties within our sample set is illustrated in Figure 2. We find that relay age, uptime, and address/port uptime can all be described by log-space normal distributions. Figure 3 shows goodness of fit between measured age distributions and the normal distributions calculated using their mean and standard deviation (σ).

Exit relays were the youngest with a mean of approximately 397 days ($\sigma \approx 467$ days). Guard relays averaged 413 days ($\sigma \approx 390$ days), and all relays which averaged 431 days ($\sigma \approx 461$ days).

Guard relays had the highest uptime average of 51 days since last restarting ($\sigma \approx 82$ days). Exit relays were second best with a 33 day average ($\sigma \approx 68$ days). Overall, relays averaged 37 days with standard deviation of 73 days.

Address or port uptime means were fairly similar across all three relay specifications. Guards averaged 250 days ($\sigma \approx 277$ days), exits averaged 250 days ($\sigma \approx 342$ days). Overall, relays averaged 233 days ($\sigma \approx 312$ days).

Figure 2. Goodness of fit of relay ages to normal distributions. For relay ages, guards actually fit better to their normal distribution than unspecified relays.

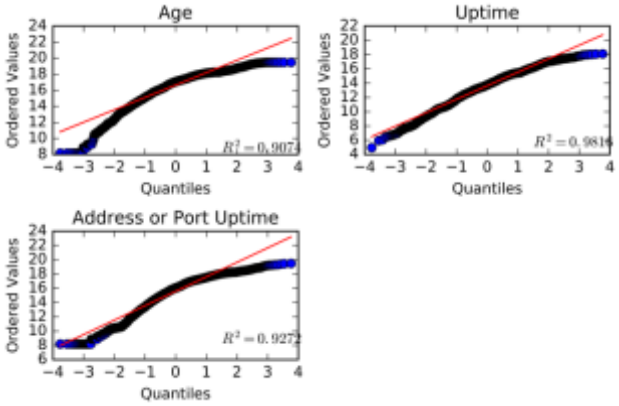
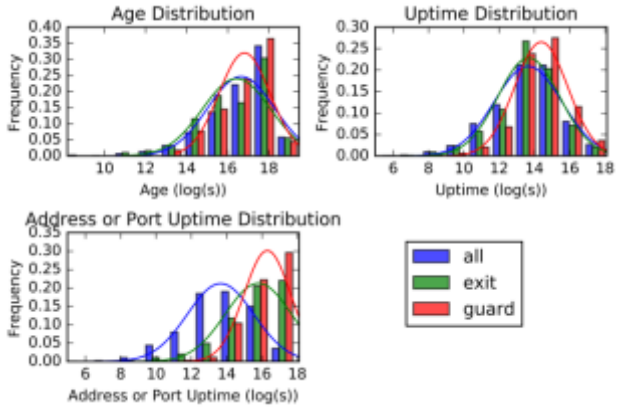


Figure 3. Distribution of timing properties in log-space.



4.1.3 Reported Bandwidths

Figure 4 compares the mean values for reported bandwidth properties in our relay dataset and Figure 5 shows a log-space distribution of the advertised bandwidth.

Figure 4. Comparison of mean reported bandwidth values separated by relay type.

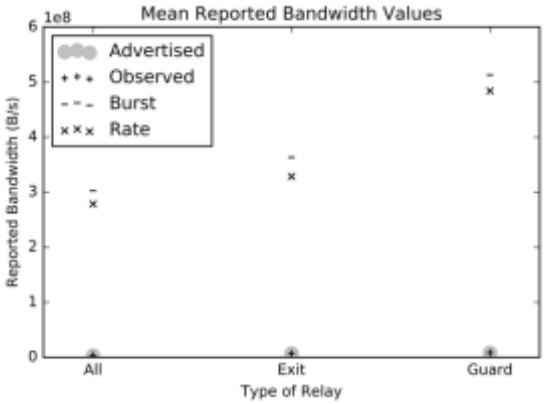
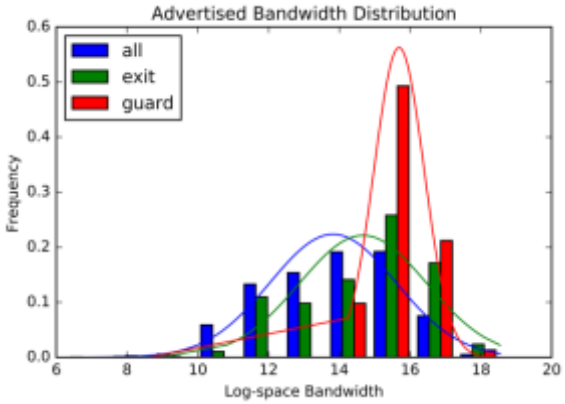


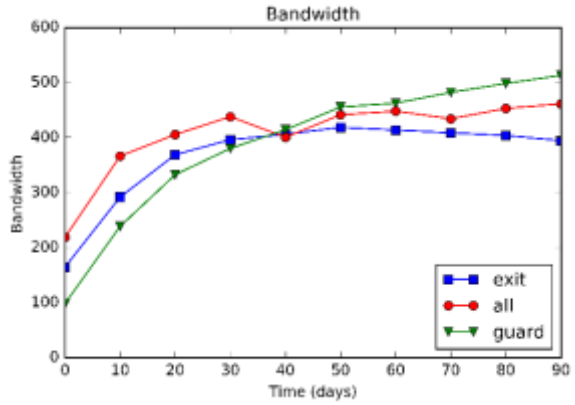
Figure 5. Distribution of advertised bandwidth.



We observe that the average relay is advertising a bandwidth of 3.3 MB/s ($\sigma \approx 5.9$ MB/s). The average exit relay is advertising a bandwidth of 6.9 MB/s ($\sigma \approx 10.1$ MB/s). The average guard relay is advertising the highest bandwidth at 8.4 MB/s ($\sigma \approx 7.6$ MB/s).

Over a 90 day period, Figure 7 shows there is a consistent upward trend in bandwidth for guard relays. Exit relay bandwidth increases steadily until about 30 days when the increase begins to slow until about 50 days after which the bandwidth actually starts to decrease. There are some interesting fluctuations in average bandwidth over time for all relays that may warrant deeper investigation.

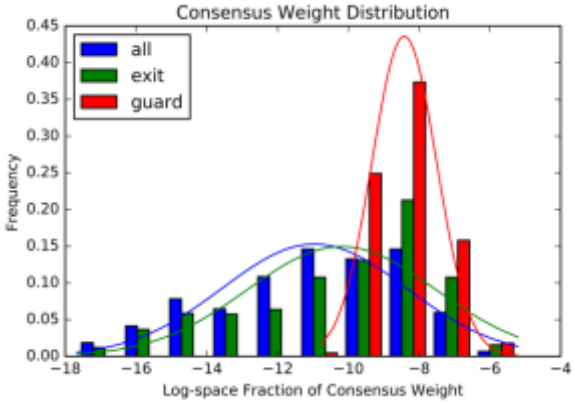
Figure 6. Change in bandwidth over time. Note that the x-axis does not represent corresponding days for each relay or relay type but rather the number of days since the relay was first observed in the consensus file.



4.1.4 *Consensus Weight*

Distribution of consensus weights can be seen in Figure 7. The average relay contributed 0.0139% ($\sigma \approx 0.0298\%$) to the total consensus weight. The average exit relay contribution of 0.0244% ($\sigma \approx 0.0450\%$), and the average guard relay had a contribution of 0.0344% ($\sigma \approx 0.0411\%$).

Figure 7. Distribution of Consensus Weights.



Consensus weight was shown to have a positive correlation with bandwidth (r=0.97) illustrated in Figure 8. This makes sense because we know from Tor dir-spec that bandwidth is in-part used to calculate consensus weight for each relay.

4.1.5 *Ports Accepted*

Distribution of the number of ports accepted is shown in Figure 9. The result is an interesting binary trend in which all relays either accept most ports or reject most ports. We also observe that exit relays with guard flags have a tendency to reject most ports.

4.1.6 *AS and Family Sizes*

Only 1379 relays from our sample set reported a list of effective family members and of these, only 99 relays also reported a list of indirect family members.

The mean effective family size (excluding relays that did not report a family) was 5.7 with a standard deviation of 7.5. The mean indirect family size was 5.2 with a standard deviation of 6.3 (excluding relays that did not report an indirect family).

There were a total of 1366 unique AS numbers contained in our relay dataset. Only 56 relays did not report an AS.

Figure 8. Correlation of advertised bandwidth and consensus weight.

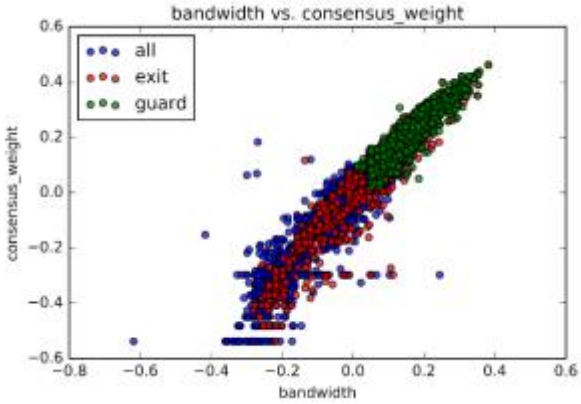
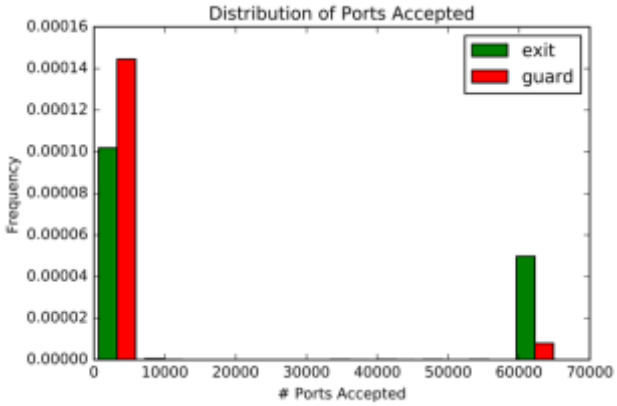


Figure 9. Distribution of ports accepted by exit relays and guard relays with the exit flag.



4.2 Guard Prediction

4.2.1 *Feature Selection*

The following features were selected for our models: advertised bandwidth, consensus weight fraction, number of ports accepted, uptime, age, and address port uptime. We also included the following flags: fast, exit, stable, v2dir. We exclude running and because 100% of relays had this flag so it would be useless as a distinguisher. We also exclude family size because our models are not impervious to missing data and only 1379 relays reported a family in their details. To include it would have meant reducing our training/testing spaces to nearly 16% of their original size. In the end, there were 7158 relays with all of the necessary properties defined.

We leave flags as binary features and convert the rest into log-space before training our model. It is important to note that this means our model learns log-space values and uses these to classify. They must therefore be converted back from log-space before any conclusions can be drawn about their comparison to real-world values.

The resulting feature set had 10 features for each relay, four binary (Fast, Exit, Stable, V2Dir) and six continuous (advertised bandwidth, age, consensus weight, ports accepted, uptime, address

port uptime). Our labels were binary with 1 = Guard, 0 = Not guard.

4.2.2 Model Performance

The worst model was our Bernoulli Naïve Bayes which was still able to accurately classify 90% of relays.

The next best models were our Logistic Regression and K-Nearest Neighbours classifiers which accurately labeled approximately 94% of relays. For K-Nearest Neighbours, we found 26 to be the optimal group size (K).

One decision tree classifier alone was able to correctly classify nearly 96% of relays with 3 as the optimal depth and 6 as the optimal number of leaf nodes.

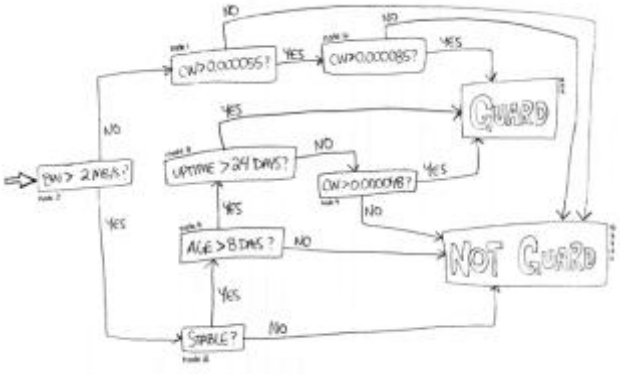
To understand the logic of our most accurate type of classifier, decision trees, we first consider the weight each feature contributes towards the final decision. Features and their weights are shown in Table 2.

Table 2. Feature Importance

Feature	Weight (%)
advertised bandwidth	33.38
consensus weight	48.00
ports accepted	0.73
uptime	4.31
age	3.08
address/port uptime	3.50
fast	2.1
exit	0.3
stable	4.12
v2dir	0.35

We then look at the decision nodes behind these features to see which decision is being made. After converting the values back from log-space, we are able to understand how each decision the tree makes is either in agreement with real-world Tor policies or not. The flow-process our decision tree uses can be seen in Figure 10. Certain decisions are directly related to real-world Tor policies. For example, at node 4, the tree decides whether or not a given relay is a guard by observing whether it is at least 8 days old. This is exactly the number of days specified in [cite].

Figure 10. Decision process of the decision tree classifier when classifying relays as guard or not. A larger version of this drawing is available at: goo.gl/1D6OrS.



5. DISCUSSION

5.1 Conclusion

We have shown that across all relays in the Tor network, most relay properties are independent of each other.

The only statistically significant correlation we found between relay properties was between relay consensus weight and bandwidth. Using Pearson's coefficient This We find that there is no statistically significant correlation between any of our collected relay properties.

We have shown that Tor guard flags can be predicted using other relay properties that can be calculated from the publicly available consensus file.

5.2 Failed Attempts

In the beginning we attempted to predict relay lifespans using regression. However, our models performed extremely poorly. We attribute this to limitations when attempting to download the Tor consensus files directly.

5.3 Future Work

Our project was limited by the fact that we were unable to download and extract relay information directly from the Consensus files. These files were too large and too frequently published. We considered downloading one sample file from each day, but this led to more complicated questions when calculating relay properties such as lifespan and uptime. It was much easier to gather these properties from the Onionoo protocol. As a result, we have no data concerning currently inactive relays. Because of this, our attempts at predicting relay lifespan were extremely hindered.

Future work may also include predicting Guard assignment in real-time.

6. ACKNOWLEDGMENTS

Our thanks to UMass Amherst for putting a café in the CS Building. This project would also not have been possible without the guidance and advice of Phillipa Gill and Rachee Singh.

7. REFERENCES

[1] "Freedom On the Net 2016." *Freedom House*. https://freedomhouse.org/report/freedom-net/freedom-net-2016.

[2] Dingledine, R., and Mathewson, N. "Tor Path Specification." https://gitweb.torproject.org/torspec.git/plain/path-spec.txt.

[3] Arma. "The lifecyce of a new relay." Blog post. *The Tor Project*. 11 Sept. 2013.

[4] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, Srinivas Devadas. "Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services", *USENIX Security '15*.

[5] T. T. Project, "Number of relays and bridges." *Tor Metrics*. https://metrics.torproject.org/servers-data.html

[6] Shadow, an open-source network simulator/emulator hybrid. https://shadow.github.io/

[7] Chris Wacek, Henry Tan, Kevin Bauer and Micah Sherr, "An Empirical Evaluation of Relay Selection in Tor", *NDSS 2013*. https://security.cs.georgetown.edu/~msherr/papers/tor-relaystudy.pdf.

[8] Anupam Das, Nikita Borisov, Prateek Mittal and Matthew Caesar, "Re 3: Relay Reliability Reputation for Anonymity Systems", *ASIA CCS 2014*. http://hatswitch.org/~nikita/papers/re3-asiaccs14.pdf

[9] Tariq Elahi, Kevin Bauer, Mashael Alsabah and Ian Goldberg, "Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor", *WPES 2012*. https://www.freehaven.net/~arma/cogs-wpes.pdf

[10] Philipp Winter, Richard Köwer, Martin Mulazzani, Markus Huber, Sebastian Schrittwieser, Stefan Lindskog and Edgar Weippl2, "Spoiled Onions: Exposing Malicious Tor Exit Relays", *PETS 2014*. http://www.cs.kau.se/philwint/spoiled_onions/pets2014.pdf

[11] Jones E., Oliphant E., Peterson P, et al., "SciPy: Open Source Scientific Tools for Python, 2001. http://www.scipy.org/.

[12] Overlier, L., and P. Syverson. "Locating Hidden Servers." *IEEE Symposium on Security and Privacy*. (2006).

[13] Pedregosa, F. et al., "SciKit-learn: Maching Learning in Python", *JMLR 12, pp. 2825-2830, 2011*.

[14] T. T. Project, "Onionoo Protocol." https://onionoo.torproject.org/protocol.html.