ANALYTICAL STUDY OF TOR HIDDEN SERVICES ON INTERNET

By

Neha Gupta/MCA/ nehaguptamca3@gamil.com

&

Harmeet Malhotra/M. Phil, MCA/ harmeet_hello@yahoo.com

Abstract

A Hidden service is a server – often delivering web pages – that is reachable only through the Tor network. While most people know that the Tor network with its thousands of volunteer-run nodes provides anonymity for users who don´t want to be tracked and identified on the internet, the lesser-known hidden service feature of Tor provides anonymity also for the server operator. Organizations run hidden services to protect dissidents, activists, and protect the anonymity of users trying to find help for suicide prevention, domestic violence, and abuse-recovery.

To track a hidden service we must first retrieve a hidden service descriptor. This is a short signed message created by the hidden service approximately every hour contain a list of introduction nodes and some other identifying data such as the descriptor id (desc id). The descID is based on a hash of some hidden service information and it changes every 24 hours. The hidden service then publishes its updated descriptor to a set of 6 responsible hidden service directories (HSDir's) every hour. These responsible HSDir's are regular node on the Tor network which have up-time longer than 24 hours and which have received the HSDir flag from the directory authorities. The set of responsible HSDir's is based on their position of the current descriptor id in a list of all current HSDir's ordered by their node fingerprint. This is an implementation of a simple DHT (Distributed Hash Table).

Keywords: Tor, Hidden service, descriptors, guard nodes, Onion routing, garlic routing

1. Introduction

Private electronic communication is becoming an increasingly important public issue. Encryption can effectively hide the content of a conversation from eavesdroppers, and this protection is being integrated into many systems. But, hiding the identities of communicating parties from eavesdroppers, or from each other, is usually not considered.

Who is communicating with whom, however, may be sensitive too. E-mail users may wish to hide their addresses. Anonymous cash is not anonymous if the communications channel identifies the purchaser. The amount of information revealed through Web browsing should be deliberate. Inter-company collaboration may be confidential. Revealing identities in a cellular phone system reveals a user's location, since the cellular phone network must track handsets' locations. Allowing the people not only to access information anonymously but also to publish anonymously is an important aspect of nurturing democracy. Similarly, providing internet services without disclosing their location and owner puts a constraint on what attacks can be performed by adversaries.

Anonymous communication networks were first introduced by David Chaum in his seminal paper [1] describing the mix as a fundamental building block for anonymity. A mix acts as a store-and-forward relay that hides the correspondence between messages it receives and sends. Several mix based architectures have been proposed and implemented to anonymise email, most notably Babel [2], Mixmaster [3] and the newer Mixminion [4]. Their latency is tolerable for email, but is unsuitable for interactive applications such as web browsing.

1.1 What is Tor

Tor stands for **T**he **O**nion **R**outer. The Tor software is a program you can run on your computer that helps keep you safe on the Internet. Tor protects you by bouncing your communications around a distributed network of relays run by volunteers all around the world: it prevents somebody watching your Internet connection from learning what sites you visit, and it prevents the sites you visit from learning your physical location. This set of volunteer relays is called the Tor network. The Tor Project is a non-profit (charity) organization that maintains and develops the Tor software.

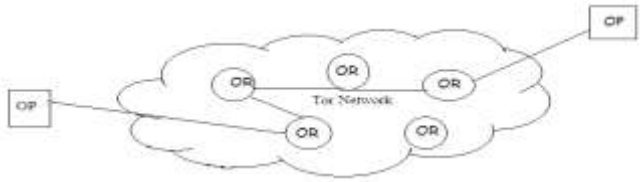

Figure 1: Tor Overview

1.2 I2P

I2P, The **I**nvisible **I**nternet **P**roject, was started in 2003 with the purpose of enabling anonymous communication in a dynamic decentralized network resilient to attacks. All communication is end-to-end encrypted and implemented as a garlic routing network layer leaving it open for use by any kind of client-server or peer-to-peer using it. The I2P developers are anonymous to the general public and only known by their pseudonyms; the founder and main developer calls himself "jrandom". The project is still in an alpha stage and is not considered mature for broad use yet. I2P has had the concept of hidden services from the beginning using a setup that is similar to the one Tor adopted. The TLD used is called .i2p and the web pages are called Eepsites.

1.3 Onion Routing and Garlic routing

Onion routing	Garlic routing
Onion routing is a technique for anonymous communication over a computer network	**Garlic routing** is a variant of onion routing that encrypts multiple messages together to make it more difficult for attackers to perform traffic analysis. The name comes from actual garlic, whose structure this protocol resembles.
Messages are repeatedly encrypted and then sent through several network nodes called onion routers.	Garlic routing is one of the key factors that distinguish I2P from Tor and other privacy/encryption networks.

Onion routing is like someone peeling an onion, each onion router removes a layer of encryption to uncover routing instructions, and sends the message to the next router where this is repeated	Garlic routing is an evolution of onion routing with changes in how messages are wrapped and routes are chosen	**1.4**
Tor uses onion routing that provides anonymous communication over the network	I2P, The Invisible Internet Project is an anonymous peer-to-peer network that uses garlic routing and was developed independently and parallel to Tor	

How to create a Tor hidden service

To create a Tor hidden service we have to follow these steps:

1. **Install Tor on your computer:** We'll have to download and install Tor on our computer. If you already have it installed, you can skip this step. By default, Tor installs the Tor browser bundle, which includes a specially configured Firefox browser. We'll see a green onion icon in our system tray when we're connected to the Tor network.
2. **Installing a web server & configuring it:** We'll need a web server to serve the hidden service site from our system. Tor recommends using the Savant web server on Windows. **Download Savant Web Server** and install it. Now open up Savant and click **Configuration.**
3. **Configure The Hidden Service:** Once you have configured your web server, now it's time to add contents to the site. So go to the place where Savant is installed, (C:\Savant\Root by default) and replace the index.html with your homepage and shut down Tor if it's running. Next, locate your torrc file. If you installed the Tor Browser Bundle, you'll find it in the *Tor Browser\Data\Tor* directory. Open this file with Notepad or another text editor.
4. Add the following section to the end of the file:

   #Hidden Service
   HiddenServiceDir C:\Users\Name\tor_service
   HiddenServicePort 80 127.0.0.1:80

5. Replace *C:\Users\Name\tor_service* with the path to a directory Tor can read and write to on your system. Do not use the directory that already contains your website. This should be an empty directory. Replace the: 80 with the port the web server is using on your system. For example, if the web server is running on port 5000, you'd use the line *HiddenServicePort 80 127.0.0.1:5000.*
6. Save the file after editing it. You'll also have to create the directory you specified, if it doesn't already exist.
7. Restart Tor after you do this. Once you have, you'll want to check the Message Log to see if there are any error messages. If the Message log is free of errors, you're good to go. Check out the hidden service directory you created. Tor will have created two files in the directory – hostname and private key. Don't give anyone the private key file or they'll be able to impersonate your hidden service Tor site.

8. Now you can use your hidden services using given hostname and private key.

1.5 Problem with Tor Hidden Service

- Anyone can set up a Tor node (HSDir: Hidden Services Directory servers) and begin logging all hidden service descriptors published to their node. They will also receive all client requests allowing them to observe the number of look-ups for particular hidden services and so the Tor node has to compromise with respect to security and reliability. There is nothing the HSDir can do to prevent these attacks; it's the responsibility of Tor Project authority to give login to authorize people.

- There is an attack on Tor where, if an Autonomous System (AS) exists on both path from Alice to entry relay and from exit relay to Bob, that AS is able to de-anonymize the path. The "bad apple attack" exploits Tor's design and takes advantage of insecure application use to associate the simultaneous use of a secure application with the IP address of the Tor user in question. One method of attack depends on control of an exit node or hijacking tracker responses, while a secondary attack method is based in part on the statistical exploitation of distributed hash table tracking.

- The list of responsible HSDir's for a hidden service is based on the calculated descriptor ID and an ordered list of HSDir fingerprints. As the descriptor ID's are predictable, and the node fingerprint is controlled by an adversary. The can position themselves to be the responsible directories for a targeted hidden service and subsequently perform a DoS attack by not returning the hidden service descriptor to clients. There is nothing the hidden service can do about this attack. They must rely on the Tor Project authority directories to remove the malicious HSDir's from the consensus

1.6 List of Tor hidden services

- **TOR Mail:** Tor Mail is a free anonymous email service provider. It is a Tor Hidden service that allows anyone to send and receive email anonymously to email addresses inside and outside the Tor network.

- **DuckDuckGo:** DuckDuckGo is using one of the hidden services. It is a search engine that emphasizes protecting searchers privacy and currently being running on the internet. It distinguishes itself from other search engines by not profiting its users and by deliberately showing all users the same search results for a given search time.

- **Silk Road:** Silk Road is an online black market. It is operated as a Tor hidden service, such that online users can browse it anonymously and securely without eventual traffic monitoring. It is used to sell drugs and other contraband.

- **Lolita City:** Lolita city was a website that used hidden services available through the Tor network. It operates through the .onion pseudo top-level domain and can be accessed via the Tor network only.

1.7 Contributions

Our implementation of anonymous connections, *onion routing,* provides protection against eavesdropping as a side effect. Onion routing provides bidirectional and near real-time communication similar to TCP/IP sockets connections [5]. Although onion routing may be used for anonymous communication, it differs from anonymous remailers [7, 10] in two ways: Communication is real time and bidirectional, and the anonymous connections are application independent. In this section we define our contribution in this paper. We study that how to do following:

• We give a method to measure the popularity of any hidden service without the consent of the hidden service operator.
• We show how connectivity to selected hidden services can be denied by impersonating all of their responsible hidden services directories.

• We demonstrate a technique that allows one to harvest hidden service descriptors (and thus get a global picture of all hidden services in Tor) in approximately 2 days using only a modest amount of resources.

• We show how to reveal the guard nodes of a Tor hidden service.

2. Work done

The first published attacks against Tor hidden services were presented by Overlier and Syverson in 2006. They targeted a previous version of the hidden services design in which no entry guard nodes were used. In the scenario described, the attacker needs to control one or more Tor relays; the idea being that given enough connection attempts, one of the attacker's relays will be chosen as the first hop of the rendezvous circuit established by the hidden service.

Valet services, improving hidden servers with a personal touch, were proposed by Overlier and Syverson again in 2006 as an extension to the hidden services concept to strengthen DoS resilience of hidden services. This is achieved by introducing an additional layer of protection for introduction points.

Another approach was presented in [7] and [8]. These attacks are based on the observation that the system clocks of computers drift depending on the temperature of the CPU. An attacker observes timestamps from a PC connected to 81 the Internet and watches how the frequency of the system clock changes when repeatedly connecting to the hidden service, causing its CPU load to rise. The attacker requests timestamps from all candidate servers and finds the one exhibiting the expected clock skew pattern. One drawback of this attack is that it assumes a closed-world model, i.e. the list of possible candidate servers needs to be known by the attacker in advance.

3. Tor Hidden Services

Tor is a low-latency anonymity network based on the ideas of onion routing and telescoping. Clients have anonymous communication to a server by proxy their traffic through a chain of three Tor relays. Specifically, prior to sending the data, a client chooses three Tor relays and uses public key cryptography to negotiate symmetric session keys with them, establishing a circuit. Whenever a client wants to send a piece of data he packs it into Tor cells and encrypts them with multiple layers of encryption using the session keys. As the cells travel along the circuit, each relay strips off one layer of encryption. Hence the server receives the original piece of data while each relay along the path knows only which relay it received the Tor cell from and which relay it forwarded the cell to.

3.1 Comparison between I2P and Tor

Tor	I2P
Tor is The Onion Router	I2P is a transport protocol comparable to IP.
Data sent in packets/datagram and Tor clients randomly determine a tunnel path for a connection	Data sent in packets/datagram and In I2P the tunnels are one way.
Tor has a concept of central directory servers for the distribution of the network	While I2P is fully distributed requiring a bootstrapping operation to find one peer to be able to join the network.
In the Tor part the exit nodes are susceptible to abuse and there are some security issues weakening the anonymity of the users.	I2P was not designed to reach regular Internet services anonymously, and there are no exit nodes in the protocol.
Uses circuit for communication	It uses Tunnel for communication
It has Entry guard nodes as well as exit node	It uses Inproxy and Outproxy
Its main part is Hidden services	Its main part is Eespsite and Destination
Between to nodes it has rendezvous point.	It has somewhat like inbound gateway and outbound endpoint
Has already solved some scaling issues I2P has yet to address	Designed and optimized for hidden services, which are much faster than in Tor
Big enough that it has had to adapt to blocking and DOS attempts	Small enough that it hasn't been blocked or DOSed much, or at all

Tor Hidden Services are a feature which was introduced in 2004 to add responder anonymity to Tor. Specifically, hidden services allow running an Internet service (e.g. a Web site, SSH server, etc.) so that the clients of the service do not know its actual IP address. This is achieved by routing all communication between the client and the hidden service through a rendezvous point which connects anonymous circuits from the client and the server.

3.2 Tor Hidden Service Architecture

The Tor hidden service architecture [9] is comprised of the following components (see Figure 1):

- **Internet** service which is available as Tor hidden service.
- **Client**, which wants to access the Internet service.
- **Introduction points (IP)**: Tor relays chosen by the hidden service and which are used for forwarding management cells necessary to connect the Client and the hidden service at the Rendezvous point.
- **Hidden service directories (HSDir):** Tor relays at which the hidden service publishes its descriptors and which are communicated by clients in order to learn the addresses of the hidden service's introduction points.
- **Rendezvous point (RP):** A Tor relay chosen by the Client which is used to forward all the data between the client and the hidden service.

Figure 1: TOR hidden services Architecture [10]

3.3 Hidden service side

In order to make an Internet service available as a Tor hidden service, the operator (Bob) configures his Tor Onion Proxy (OP) which automatically generates new RSA key pair. The first 10 bytes of the SHA-1 digest of an ASN.1 encoded version of the RSA public key become the identifier of the hidden service. The OP then chooses a small number of Tor relays as introduction points and establishes new introduction circuit to each one of them (step 1 in Figure 1).

As the next step (step 2), Bob's OP generates two service descriptors with different IDs, determines which hidden services directories among the Tor relays are responsible for his descriptor and uploads the descriptor to them. A hidden services directory is a Tor relay which has the HSDir flag. A Tor relay needs to be operational for at least 25 hours to obtain this flag. The hidden service

descriptors contain the descriptor ID, the list of introduction points and the hidden service's public key.

3.4 Client side

When a client (Alice) wants to communicate with the hidden service, she needs a pointer to this service, which needs to be transmitted out of band. The pointer is the hostname of the form"z.onion", where z is the base-32 encoded hidden service identifier described above. She then computes the descriptor IDs of the hidden service (see the expression in section III-C) and the list of responsible hidden service directories and fetches the descriptors from them (step 3).

In order to establish a connection to a given hidden service Alice's OP first builds a rendezvous circuit (step 4). It does this by establishing a circuit to a randomly chosen Tor relay (OR), and sending a RELAY_COMMAND_ESTABLISH_RENDEZVOUS cell to that OR. The body of that cell contains a Rendezvous cookie (RC). The rendezvous cookie is an arbitrary 20-byte value, chosen randomly by Alice's OP.

Alice chooses a new rendezvous cookie for each new connection attempt. Upon receiving a RELAY_COMMAND_ESTABLISH_RENDEZVOUS cell, the OR associates the RC with the circuit that sent it. Alice builds a separate circuit to one of Bob's chosen introduction points, and sends it a RELAY_COMMAND_INTRODUCE1 cell containing the IP address and the fingerprint of the rendezvous point, the hash of the public key of the hidden service (PK ID), and the rendezvous cookie (step 5).

If the introduction point recognizes PK ID as the public key of a hidden service it serves, it sends the body of the cell in a new RELAY_COMMAND_INTRODUCE2 cell down the corresponding circuit (step 6).

When Bob's OP receives the RELAY_COMMAND_INTRODUCE2 cell, it decrypts it using the private key of the corresponding hidden service and extracts the rendezvous point's nickname as well as the rendezvous cookie. Bob's OP builds a new Tor circuit ending at Alice's chosen rendezvous point, and sends a RELAY_COMMAND_RENDEZVOUS1 cell along this circuit, containing RC (step 7). Subsequently, the rendezvous point passes relay cells, unchanged, from each of the two circuits to the other.

In this way, the client knows only the rendezvous point. Neither does the hidden service learn the actual IP address of the client nor does the client learn the IP address of the hidden service.

3.6 Choosing responsible HSDir's

A hidden service determines if a hidden services directory is responsible for storing its descriptor based on the descriptor's ID and the directory's fingerprint [11]. Descriptor identifiers change periodically every 24 hours and are computed as follows:

descriptor-id = H(public-key-id || secret-id-part) secret-id-part = H(descriptor-cookie || time-period || replica-index)[14]

The field descriptor-cookie is an optional field. If present, it prevents non-authorized clients from accessing the hidden service. The field *time period* denotes the number of days since the epochs. This is used to make the responsible directories change periodically. The *replica index* is used to create different descriptors identifiers so that the descriptor is distributed to different parts of the fingerprint range.

After computing the descriptor identifiers, a hidden service determines which directory nodes are responsible for storing the descriptor replicas. To do this the hidden service arranges the directories using their fingerprints in a closed fingerprint circle and chooses as hidden service directories the three closest relays in positive direction (fingerprint value of them is greater than the fingerprint value of the hidden service).

3.6 Guard nodes

Being a low-latency anonymity network Tor is vulnerable to the traffic confirmation attacks: if an adversary can monitor the edges of a Tor circuit, she can confirm who is communicating. This is quite dangerous for hidden services since by design the attacker always controls one edge of the connection. If the entry nodes of the circuit were chosen uniformly from the whole set of Tor relays, the probability of the attack would approach 1 when the number of circuits to the hidden service established by the attacker increased. In order to significantly reduce the probability of the traffic confirmation attack Tor developers introduced the concept of *entry guard nodes*. Tor initially selects a set of three guard nodes. Whenever less than two guard nodes from the set are reachable, new guard nodes are chosen. A guard node remains in the set for a random duration between 30 and 60 days. Then it is marked as expired and removed from the set. Whenever a circuit is established, one node from the set of Guard nodes is used for the first hop.

4. Transition of Data & Decoding of Descriptors

In this section we study the security of descriptor distribution procedure for Tor hidden services. We show how an attacker can gain complete control over the distribution of the descriptors of a particular hidden service. This undermines their security significantly: before being able to establish a connection to a hidden service, a client needs to fetch the hidden service's descriptor; unless it has it cached from a prior connection attempt. Thus, should the attacker be able to control the access to the descriptors, the hidden service's activity can be monitored or it can be made completely unavailable to the clients.

4.1 Controlling hidden service directories

As mentioned in the background section, the list of responsible hidden service directories depends on the current consensus document and the descriptor IDs of the hidden service. In this subsection, we explain how to inject relays into the Tor network that become responsible for the descriptors of the hidden service. This immediately translates into the problem of finding the right public keys, i.e. the keys with fingerprints which would be in-between the descriptor IDs of the hidden service and the fingerprint of the first responsible hidden service directory.

As a proof of concept we used this approach to control one of the six hidden service directories of the discovered Tor botnet, the Silk Road hidden service, and the DuckDuckGo hidden service. We tracked these for several days and obtained the following measurements:

(1) The number of requests for the hidden service descriptor per day (see Tables I and II) &
(2) The rate of requests over the course of a day, which is shown in Figure 2 (each point corresponds to the number of hidden service descriptor requests per one hour).

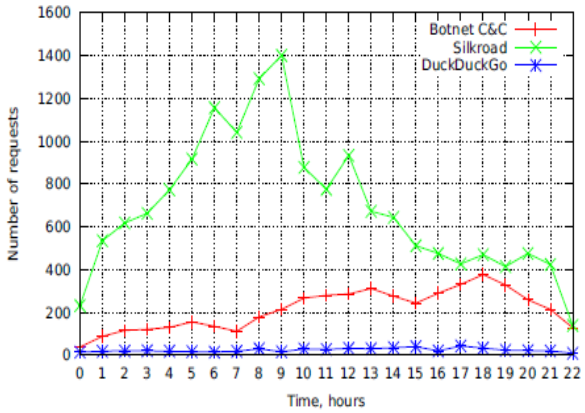

Figure 2: Hidden service descriptor request rate during one day.[Alex Biryukov, Ivan Pustogarov, Ralf-Philipp Weinmann: Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization in 2013 IEEE Symposium on Security and Privacy]

Column 1 of Table I and columns 2 and 4 of Table II show the number of requests for a particular hidden service descriptor per day [12]. Columns "Total" show the total number of descriptors requests (for any hidden services descriptor) served by the hidden service directory per day. The hidden service tracked in Table I is the IRC C&C service.

Table 1Popularity of Discovered Botnet[Alex Biryukov, Ivan Pustogarov, Ralf-Philipp Weinmann: Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization in 2013 IEEE Symposium on Security and Privacy**]**

Date	Botnet Descriptor	Total
13 Jul	1408	6581
14 Jul	1609	2392
15 Jul	1651	4715
16 Jul	1448	6852
25 Jul	4004	6591

26 Jul	4243	4357
27 Jul	4750	4985
28 Jul	4880	7714
29 Jul	4977	9085

Table 2
Popularity of SILK ROAD and DUCKCUCKGO[Alex Biryukov, Ivan Pustogarov, Ralf-Philipp Weinmann: Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization in 2013 IEEE Symposium on Security and Privacy**]**

Date	Silk Road	Total	DuckDuckGo	Total
09 Nov	19284	27363	502	2491
10 Nov	15427	16103	549	5621
11 Nov	15185	15783	543	3899
12 Nov	15877	16723	549	10910

5. Deanonymisation Of Hidden Services

The fact that an attacker always controls one side of the communication with a hidden service means that it is sufficient to sniff/control a guard of the hidden service in order to implement a traffic correlation attack and reveal the actual location of the hidden service. In particular, an attacker can:

- Given the onion address of a hidden service with unencrypted list of introduction points determine if her guard nodes are used by this hidden service.
- Determine the IP addresses of those hidden services that use the attacker's guard nodes.
- Determine if the attacker's guard nodes are used by any of the hidden services, even if the list of introduction points is encrypted.

5.1 Unencrypted descriptors

In order to confirm that an attacker controls a guard node of a hidden service she needs to control at least one more Tor non-Exit relay. In the attack, the hidden service is forced to establishes rendezvous circuits to the rendezvous point (RP) controlled by the attacker. Upon receiving a **RELAY_COMMAND_RENDEZVOUS1** cell with the attacker's cookie, the RP generates traffic with a special signature. This signature can be identified by the attacker's middle node. We note that a special **PADDING** cell mechanism in Tor simplifies generation of a signature traffic which is

discarded at the recipient side, and is thus unnoticeable to the hidden service. The steps of the attack are shown in Figure 3 and are as follows:

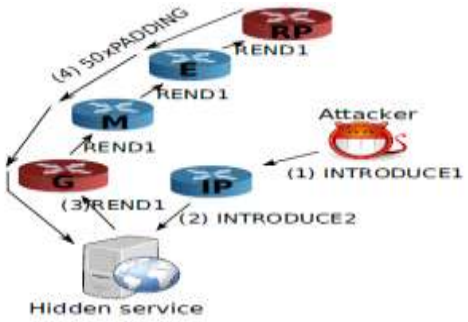

Figure 3: Revealing the Guard nodes[Alex Biryukov, Ivan Pustogarov, Ralf-Philipp Weinmann: Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization in 2013 IEEE Symposium on Security and Privacy]

1. The attacker sends a RELAY_COMMAND_INTRODUCED1 cell to one of the hidden services's intorduction points (IP) indicating the address of the rendezous point.
2. The intorduction popint forwads the content in a RELAY_COMMAND_INTRODUCE2 cell to the hidden service.
3. Upon receving the RELAY_COMMAND_INTRODUCE2 cell, the hidden service estsblishes a three-hop circuit to the indicated rendezovs point and sends it a RELAY_COMMAND_RENDEZVOUS1 cell.
4. When the rendezvous point controlled by the attacker receives the RELAY_COMMAND_RENDEZVOUS1 cell, it sends 50 PADDING cells back along the rendezvous circuit which are then silently dropped by the hideden services.
5. The rendezvous points sends a DESTROY cell down the rendezvous circuit.

Whenever the rendezvous point receives a RELAY_COMMAND_RENDEZVOUS1 with the same cookie as the attacker sent in the RELAY_COMMAND_INTRODUCTION1 cell it logs the reception. At the same time, the attacker's guard node monitors the circuits passing through it. Whenever it receives a DESTROY cell over a circuit it checks:

1) Whether the cell was received just after the rendezvous point received the RELAY_COMMAND_RENDEZVOUS1 cell,
2) The number of the forwarded cells: 3 cells up the circuit and 53 cells down the circuit. Three cells more come from the fact that the hidden service established a circuit to the rendezvous point thus the attacker's guard node had to forward (2×RELAY_COMMAND_EXTEND + 1×RENDEZVOUS1) cells up and (2×RELAY_COMMAND_EXTENDED + 1×DESTROY) cells down. This is very important for our traffic signature since it allows us to distinguish the case when the attacker's node was chosen as the guard from the case when it was chosen as the middle.

If all the conditions are satisfied, the attacker decides that her guard node was chosen for the hidden service's rendezvous circuit and marks the previous node in the circuit as the origin of the hidden service. In order to estimate the reliability of the traffic signature, we collected a statistics on the number of forwarded cells per circuit. We examined 748,846 circuits on our guard node. None of

the circuit exhibited the traffic pattern of 3 cells up the circuit and 53 cells down the circuit. This means that the proposed traffic signature is highly reliable.

5.2 Encrypted descriptors

If the list of introduction points is encrypted, an attacker will not be able to establish a connection to the hidden service. Hence the attack described in the previous section does not apply. However, we can use a different method to determine if some of those encrypted hidden services use a guard node controlled by us. We will not be able distinguish between hidden services with encrypted introduction points though. On the other hand, note that results from Section IV show that the number of hidden services which encrypt their introduction points is comparatively small. To achieve this goal we do the following:

- On our guard node we look for a traffic pattern characteristic for introduction circuits.
- We discard introduction circuits [13] which originate at the same IP address as any of the hidden services with unencrypted descriptors.
- For all remaining introduction circuits, we mark their origins as possible locations of encrypted hidden services.

6. Conclusions

Tor hidden services were originally implemented as some a simple feature on top of the Tor network. There are discussions under way to re-implement hidden services to allow them to scale more efficiently. We analyzed the security properties of Tor hidden services and shown that attacks to deanonymize hidden services at a large scale are possible with only a reasonable amount of resources.

In this paper we also reveal the guard node of hidden services. We can harvest guard nodes and descriptors by using relay and rendezvous points etc. commands. Note that the above situation is nothing more than substitute measures. We believe that the problems that we have shown are serious enough to justify a careful redesign of Tor hidden service.

7. References

[1] ANONYMOUS. IAmA a malware coder and botnet operator, AMA. http://www.reddit.com/r/IAmA/comments/sq7cy/ iama a malware coder and botnet operator ama/, April 2012.

[2] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–88, 1981.

[3] Improving Performance and Anonymity in the Tor Network Andriy Panchenko, Fabian Lanze, and Thomas Engel Interdisciplinary Centre for Security, Reliability and Trust (SnT) University of Luxembourg

[4] SCHUMER, C. E., AND MANCHIN, J. Press release: Manchin urges federal law enforcement to shut down online black market for illegal drugs. http://manchin.senate.gov/public/index.cfm/2011/6/manchinurges- federal-law-enforcement-to-shut-down-online-blackmarket- for-illegal-drugs, Jun 2011.

[5] CHRISTIN, N. Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. *CoRR abs/1207.7139* (2012).

[6] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT- 22(6):644–654, 1976.

[7] THE TOR PROJECT. Tor Rendezvous Specification. https://gitweb.torproject.org/torspec.git?a=blob plain; hb=HEAD;f=rend-spec.txt, accessed on November 13th, 2012.

[8] ELAHI, T., BAUER, K., ALSABAH, M., DINGLEDINE, R.,AND GOLDBERG, I. Changing of the guards: A framework for understanding and improving entry guard selection in Tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2012)* (2012), CM, pp. 43–54.

[9] ELICES, J., PEREZ-GONZALEZ, F., AND TRONCOSO, C.Fingerprinting Tor's Hidden Service Log Files Using a Timing Channel. In *WIFS 2011 – 3rd IEEE International Workshop on Information Forensics and Security* (2011), IEEE.

[10] BIRYUKOV, A., PUSTOGAROV, I., AND WEINMANN, R.P. TorScan: Tracing long-lived connections and differential scanning attacks. In *ESORICS 2012*, S. Foresti, M. Yung, and F. Martinelli, Eds., vol. 7459 of *LNCS*. Springer, 2012, pp. 469–486.

[11] M. Rennhard and B. Plattner. Introducing MorphMix: Peerto Peer based Anonymous Internet Usage with Collusio Detection. In *Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[12] I. S. Moskowitz, R. E. Newman, D. P. Crepeau, and A. R. Miller. Covert channels and anonymizing networks. In *Workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, October 2003.

[13] LIPMAN, D. H. Mailing list post: "[tor-talk] vwfws4obovm2cydl.onion ?". https://lists.torproject.org/ pipermail/tor-talk/2012-June/024565.html, June 2012.

[14] Alex Biryukov, Ivan Pustogarov, Ralf-Philipp Weinmann: Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization in 2013 IEEE Symposium on Security and Privacy