FAKULTÄT FÜR !INFORMATIK

Anonymity & Monitoring

How to Monitor the Infrastructure of an Anonymity System

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computer- und Datensicherheit

ausgeführt von

Martin Mulazzani

Matrikelnummer 0225055

an der:

Fakultät für Informatik der Technischen Universität Wien

Betreuung:

Betreuer: Prof. Dr. A Min Tjoa

Mitwirkung: Univ.-Ass. Dr. Edgar Weippl

Wien, 29.01.2009

Unterschrift Verfasser

Unterschrift Betreuer

Technische Universität Wien

A-1040 Wien Karlsplatz 13 Tel. +43/(0)1/58801-0

http://www.tuwien.ac.at

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 29.01.2009

Unterschrift

Abstract

This work is about monitoring the infrastructure of the TOR network. TOR, “The Onion Router”, is an anonymity network that is widely used in the world to hide the user’s IP address. It is used by an estimated tens of thousands of users every day. The infrastructure is run by volunteers. More than 1.000 servers are running all the time, distributed all over the world.

To detect attacks on the TOR network and to provide feedback to the user about the current state of the network, a monitoring of the TOR servers has been implemented. By collecting data over time this monitoring can be used in the future for further improvements in the TOR network or the client software. Additionally to this, the current state of anonymous Internet communication is presented, a task which has received a lot of attention in recent years by the scientific community.

Over a period of three months this monitoring data has been collected to show the current state of the network. Interesting informations and patterns have been extracted and are presented. The most interesting pattern: the TOR network is very stable, more then half of all the servers are from Germany and the USA and a daily sine pattern influences the number of servers.

The implementation was added to the official TOR Status project and is released under an open source license.

Kurzfassung

Diese Arbeit befasst sich mit dem Thema wie ein Anonymisierungsdienst im Internet überwacht werden kann, ohne dabei die Anonymität der Nutzer zu gefährden. Im speziellen geht es um das Überwachen des TOR Netzwerkes, "The Onion Router". TOR ist ein Anonymisierungsdienst der die IP Adresse der Benutzer verschleiert. Geschätzte Zehntausende Benutzer verwenden es pro Tag. Die zugrunde liegende und benötigte Infrastruktur wird von Freiwilligen betrieben, wobei mehr als 1.000 TOR Knoten ständig online sind. Diese sind über die ganze Welt verteilt.

Es wurde eine Überwachung des TOR Netzwerkes implementiert, mit dem Ziel Angriffe leichter erkennbar zu machen und den Benutzern ein Feedback über den aktuellen Status des TOR Netzwerkes zu geben. Diese Überwachung kann in Zukunft dafür verwendet werden das TOR Netzwerk und die Benutzersoftware zu verbessern. Zusätzlich wird der aktuelle Status von anonymer Kommunikation im Internet erläutert.

Über einen Zeitraum von drei Monaten wurden die Überwachungsdaten gesammelt. Interessante Muster und Informationen werden aufgezeigt. Die interessantesten Muster: das TOR Netzwerk ist sehr stabil, mehr als die Hälfte aller Server kommen aus Deutschland und Amerika und es gibt ein tägliches, sinusartiges Muster in der Anzahl der Server.

Die Implementierung wurde dem offiziellen TOR Status Projekt hinzugefügt und ist unter einer Open Source Lizenz veröffentlicht.

Contents

1 Introduction

Todays Internet is a dangerous place. The normal user is confronted with Worms, Viruses, Malware and Spyware and other threats on a daily base. Malicious software for example tries to capture valuable user data like online bank informations and logins or credit card numbers. There are special threats with respect to information flows, e.g. tracing the information exchange, capturing the information, altering information during transmission, redirecting to unintended recipients, or manipulating the information a user gets thus manipulating the user himself.

To address these threats, we have to take a step back and look at the broader picture. The Internet is build around the exchange of messages. These messages can be of various forms. On application layer there is e.g. the transportation of emails or the access to websites. On transportation layer this is mapped to data packets transporting this information. It can be voice communication or bank transactions, something supposed to be confidential (except for the communicating parties). These messages can contain a lot of information, useful in many different aspects and for different parties. By intercepting these messages or altering them, hostile parties can control and use this flow of information for their needs. These parties can be countries or governments, who want to control what their citizen do on the Internet, it can be Internet service provides trying to protect their users from malicious content or fraud, or spies endangering national security or aggressive companies trying to profile users in order to sell this information or to misuse this information. However, the user often is not even aware that something is happening with the flow of information.

This can be prevented by the use of anonymous communication methods.

Anonymity on the Internet is a relatively young field of research. The first anonymity system introduced was using long delays until the message reached its destination to ensure anonymity. But as the Internet was not in widespread use that time, its broad usage never occurred. In recent years anonymity systems like TOR or JAP have been designed and deployed to allow low latency anonymous communication for everyone. This work is focusing on TOR, and how simple monitoring can be used without endangering anonymity.

This work is organized as follows: The second chapter discusses basic terminologies and principles of anonymity along with current methods of

Internet manipulation, principles of anonymous communication and possible attacks on anonymity sytems. In chapter 3 the well deployed anonymity system TOR is described, its architecture as well as alternative anonymity systems. Chapter 4 presents the approach chosen for the monitoring of the TOR network, the basic conflict between anonymity and monitoring, the selected values for the monitoring, the associated anonymity aspects and the implementation which has been realized. Chapter 5 discusses the collected data and what patterns have been found, observed regular patterns as well as strange patterns which should receive further attention. The last chapter is dedicated to the overall conclusion.

2 Fundamentals

2.1 Anonymity

The word anonymity comes from the Greek language, meaning “without a name” or “namelessness”. When talking about anonymity, the usual setting refers to the transportation of messages from senders to recipients [39]. These messages can be of any kind (like electronic messages, votes in general elections, emails, ...) and can be transported using any media (a piece of paper, radio waves, the Internet, natural language, ...).
Technically it is possible that the senders of a message remain anonymous whereas the recipients are well known (for example an anonymous email send by you to your boss), the recipients are anonymous and the senders are well known (your boss replies to your email without knowing who you are), or both (you send an email to someone without ever finding out that the recipient is actually your boss).

“Anonymity is the state of being not identifiable
within a set of subjects, the anonymity set”

The senders and the recipients each have their own anonymity set, which might be disjoint, overlap or be the same [40]. These sets can change over time, as new subjects can join and others can leave the sets. The larger the anonymity set is and the more evenly sending and receiving is distributed, the stronger is the resulting anonymity.

There exist 3 different kinds of anonymity regarding message transportation:

- *Sender anonymity*: a message is not linkable to a particular sender, and a particular sender is not linkable with any messages.
- *Recipient anonymity*: a message is not linkable to a particular recipient, and a particular recipient is not linkable with any messages.
- *Relationship anonymity*: it is untraceable who communicates with whom. This is a weaker property then sender or recipient anonymity as it might be known who sends or receives messages, but not the communication relationship.

Unlinkability is the more general term for this, as it means that two items of interest (like subjects, messages or any combination of those) are from the attackers perspective no more or less related after the interaction than they were before.

Another important property of anonymity is *unobservability.* This means that it is not possible for third parties to tell if a communication medium is being used at all, respectively if somebody is sending or receiving messages. This requires that the messages are not distinguishable from random noise.

Pseudonyms or *aliases* are identifiers without revealing the real identity of the subject [40]. There are different kind of pseudonyms, depending on the degree of linkability and application:

- Person pseudonym: A person is identified by a pseudonym. This is not necessarily a 1:1 relation, as a person might be the holder of more then one pseudonyms.
- Group pseudonym: If a pseudonym is shared by many persons, this set of people becomes a group.
- Role pseudonym: Sometimes it is sufficient to be identified by a role, which might be assigned by a third party.

All forms of pseudonyms are used frequently in connection with computers. Examples for the everyday use of pseudonyms include computer logins, emails, public key cryptography and web applications like eBay or Wikipedia.

In other fields of computer science the term anonymity is used differently. For example in connection with databases and data publishing, when statistical data is published, privacy related and sensitive data like names get removed to ensure privacy. Sweeney [48] defines the term "k-anonymity" which means that in every dataset published every tuple occurs at least k times. Thus every tuple has an anonymity set of size k. It is based on the observation that pseudonymized data sets like medical records or voter registrations can be linked to identify the corresponding individual even though the identifying attributes like name, address or telephone number have been removed in both datasets. If k=2 this is sufficient to prevent the unique identification based on e.g. gender, date of birth and postal code, which is often published in such datasets. However, this has been shown to be a NP-hard problem [27].

2.2 Content and Communication Blocking

Every flow of information has some kind of value, at least for the communicating parties. Third parties might find it valuable if the could stop or alter the flow of information, especially for example in warfare intelligence. We will not go into details how Internet communication works, but there exist various methods of communication and data transfer blocking on the Internet. Some common examples which are quite easy to implement and bypass include:

Block on IP Address Every IP packet transmitted on the Internet has a source and a destination IP address. For Internet Service Providers it is rather easy to block access to certain addresses by dropping every packet destined to it. The other way around, if a computer runs riot and sends malicious, malformed packet somewhere e.g. as part of a distributed denial of service attack, border routers can easily be configured to drop all or special packets by that IP address. Notice, however, that multiple users might share an IP with network address translation at anytime, and multiple services like different websites might run on a single IP, which makes this kind of blocking vulnerable to collateral damage (which might of course be acceptable in certain situations). For details on the Internet Protocol see the official RFC by the Internet Engineering Task Force [43].

DNS Poisoning and Redirection Instead of blocking the IP address of a computer, a more fine grained attack is to block or alter the domain name resolution. Normally this happens not countrywide, but at the Internet Service Providers as most of them run their own DNS server. The blocking can be done either by simply dropping all requests for a certain domain name rendering it inaccessible, sending bogus answers with IP addresses which do not exist or are not reachable or redirect the request to another IP address than the valid one. This can be used to display a warning page, e.g. that the entered URL is suspect to be a phishing website.

DNS Poisoning is often used if a country wants a website ad-hoc blocked for all of its citizen. Recent examples are:

- Italy - thepiratebay.org: In August 2008 an Italian court ordered that the website of thepiratebay.org, a Swedish BitTorrent tracker, has to be blocked. Alleged 70 % of the Italian providers followed court order and blocked the domain name and the IP addresses belonging to thepiratebay.org. This was not very effective, as the website operators simply registered a different domain (labaia.org) and new IP addresses, which rendered the filters useless. At the end of September 2008, another court ordered that the blockade was illegitimate.

- China - many websites: DNS poisoning is one of many techniques used behind the “Great Firewall of China”, which is discussed in detail in section 2.2.1.

- Turkey - youtube.com: Several times in the last years, youtube.com was blocked by Turkish Internet provides due to court orders because of improper video files, e.g. one “offending” the founder of the Republic of Turkey, Mustafa Kemal Atatürk. Youtube is a video platform where users can put their self made video clips online and is one of the most visited websites in the world.
- Brazil - wordpress.com: In April 2008, a Brazil court ordered to block access to a certain weblog hosted at wordpress.com. As the Internet service providers in Brazil had no access to sophisticated filtering to block one specific weblog, the whole domain got blocked. Other free weblog hosting websites like blogspot.com are blocked in various countries like China and Iran.
- other countries - youtube.com: As video and audio content is much harder to filter video platforms are a primary target for censorship in countries. Many countries block youtube.com by default, for example Tunisia, China, Syria, Sudan or Iran. Other countries block or blocked it temporarily. In Saudi Arabia the youtube.com website works normally, but the webpage is blocked where you can enter your date of birth to confirm that you are old enough to view videos flagged for adults only. Thus all adult flagged videos are not viewable in Saudi Arabia.

To bypass DNS Poisoning and redirection, alternative and open DNS servers can be used. One example is OpenDNS, which is a free worldwide DNS resolution service. For details on the Domain Name System see the official RFC by the Internet Engineering Task Force [30].

Block on Packet Content As long as TCP packets are not encrypted with TLS or any other encryption method it is possible to filter content on certain keywords. These keywords can be in the requested URL or in the data section of packets. This technique is heavily used to filter the Internet e.g. in China (see 2.2.1). Just dropping the affected packages would not be sufficient, as they will get retransmitted by the Transmission Control Protocol and might take another route to the destination, bypassing the filtering device. The connection needs to be closed e.g. by a forged packet with the reset flag set which tears down immediately the TCP connection. For details on the Transmission Control Protocol see the official RFC by the Internet

Engineering Task Force [7].
Another possible approach is to force the users to use a proxy server. This technique has the advantage that the list of keywords is easily adaptable by human operators for new keywords and the blocking is very precise and without much collateral damage. But compared to other techniques proxy servers have quite heavy resource demands, making it impracticable for widespread use. Many systems use a Proxy server which simply rejects to serve certain requests. According to the OpenNet initiative this is in use for example in Burma and Saudi Arabia, whereas China uses the intrusion detection approach with forged TCP reset packets.

Block on Application Layer Of course, every application on the Internet can be used for further blocking. One example is Google, which censors right wing and Nazi material links in their search engine in central Europe. In France and Germany e.g. the website stormfront.org is not among the search results for appropriate searches. When Google launched the localization of the Google search engine in China 2006 (www.google.cn), many websites were and still are censored by the Chinese government. Before that time, all search queries in China had to leave the country and therefore pass the "Great Firewall of China", unwanted queries were filtered there. But since then, Google takes care of the filtering as all queries are processed without traversing the great firewall. This content blocking is not based on the above techniques which operate on the network layer, but because of manipulation of the ranking of websites at the application layer, respectively because of manipulations and removals directly in the heart of the Google search engine, in the so called Google index.
But other search engines do the same to be able to offer their service in China, for example Microsoft and Yahoo. Yahoo additionally gives personal information from their mail users to the Chinese authorities if they request it, as happened e.g. to the Chinese reporter Shi Tao in 2004. He sent a confidential document from the communist party about censorship and the Tienanmen massacre to an overseas democracy web site called Asia Democracy Foundation and was sentenced to prison for 10 years. Other dissidents that were sentenced to jail because of Yahoo are Li Zhi, Jiang Lijun, and Wang Xiaoning, with an unknown number of unpublished cases.
Another example for application layer blocking got recently public, also in China: Skype. Skype is a free Voice over IP client, which transmits voice communication, instant messages or files encrypted between clients and is freely available. The Chinese version of Skype, called TOM-Skype, has the additional "feature" to filter text messages containing words like "Tibet",

"Taiwan", "Falun Gong" or "communist party" which are considered dangerous by the Chinese government. It also used to save those messages (and might still do) including personal user information on an insecure server, which got hacked and the data revealed by the Citizen Lab from the University of Toronto [54]. The servers were insecure and contained the passwords to decrypt the saved messages. Users using the non-Chinese version were also affected if they exchanged instant messages with a user using the TOM-Skype version. It is unknown who has access to this data or what it is used for.

2.2.1 The Great Firewall of China

The so called "Great Firewall of China", also known as the "Golden Shield Project", is a surveillance and censorship system for the Internet in the People's Republic of China. Many websites which are related to "unwanted content" or are affiliated to unwanted political groups or ideologies are blocked at the routers connecting China with the rest of the world. The exact set of blocked websites changes continuously, but good examples are some pages of Wikipedia (e.g. Dalai Lama and Falun Gong) and the websites of Amnesty International and Human Rights Watch. Because of the Olympic Games in Beijing in 2008 and the pressure from the international press and the International Olympic Committee, many sites were unblocked.

Back in May 2006, blocking was done in part by inspecting TCP packets for keywords. As the Chinese Government is not publishing any technical information on the system, this was reverse engineered by Steven Murdoch et al. [44]. If this system detects a malicious keyword, it resets the connection by sending forged TCP packets with the RST flag set to both communication partners. This happens symmetrical for both communication directions, as the content should be blocked entering China as well as leaving it. Furthermore, the connection is blocked for any other data up to 1 hour (even if no keyword is used) by further forged resets. Other techniques used include DNS redirection (the firewall sends a spoofed DNS reply with a wrong or different IP address), filtering based on words in the URL and forged SYN/ACK packets during the TCP three way handshake [56]. If the content of the packet is encrypted, keyword matching becomes impossible and even content with the keywords can pass through the firewall, if no other method stops the communication.
By manipulating the TCP stack to ignore all packets with the RST flag set, Murdoch et al. were able to bypass the firewall completely as this is (or

was) apparently the main way of blocking data transfers. Of course, both communication parties (inside and outside of China) need the manipulated TCP stack, as reseting one side of the communication is enough to stop data transfers. This approach has practical drawbacks, as many firewalls start filtering by IP address once a sufficient number of packets with the RST flag set have been sent [14]. They even proposed a way for using the firewall for Denial of Service attacks by sending forged packets containing one of the keywords through the firewall. This works only on a pair of IP addresses, but might still be enough for an attacker in some cases.
The TOR Project uses another approach to increase blocking resistance, so called bridges [14]. Bridges are computers that are not regular TOR server and are therefore not listed in the directory, but can be used to access the TOR network. The big advantage is that it becomes much harder to collect informations about all available bridges, as there exists no public complete list of all bridges and the users behind restrictive filtering systems only need one bridge to bypass the filter. Details on TOR bridges can be found in section 3.1.1.

In the near future the Chinese Internet and how it is filtered will remain a topic worth tracking. According to internetworldstats.com, 250 million Chinese had access to the Internet in 2008, which is about only 19 % of the population but already more in total compared to the United States of America (215 million). The number of users will continue to grow rapidly, and the traffic handled by the firewall too due to increasing spread of multimedia content like pictures or videos. This will also be very interesting, as it is quite easy and straight forward to automatically block connections based on keywords, but rather difficult for images and audio/video content.

2.3 Anonymity on the Internet

Anonymity on the Internet is not trivial. Both communicating computer systems need to know how to address the communication partner or where the message is going to, so that the routing systems know how to route it. By adding intermediate nodes on the path and encrypting the message, the original sender can protect his data and his communication. But what to protect: the user, the application used to send the message or the current IP address of the computer?

All anonymity systems make a compromise between flexibility and anonymity. The system can be very flexible, so that many applications are able use it. But then, if many different applications use it it becomes easier to detect application patterns or communication patterns. Browsing the web has a different pattern as sending an email, these patterns can act like application fingerprints. If the system is application or protocol specific, it provides great anonymity but it is not very flexible. Thus the anonymity can work on different layers of communication. For browsing the web, this can be either done on the Internet layer (IP), the transport layer (TCP) or the application layer (HTTP). HTTP is very specific, while IP is very flexible and usable for many other things. Depending on the threat model of the anonymity system, this is an important decision. An early anonymity system, "onion routing", which was the predecessor of TOR, used application anonymity but failed due to the lack of flexibility [19, 20].

Anonymous electronic communication was first introduced in the paper "Untraceable electronic mail, return addresses, and digital pseudonyms" in 1981 by David Chaum [9]. Instead of sending the message directly from the sender to the recipient, it passes several additional relay servers on its way (so called mixes). These mixes collect the messages and forward them in batches. With the use of public key cryptography [13] it is possible that only the intended receiver can read the message and not the mixes. To prevent the mixes from reading the (encrypted) messages they get additionally encrypted before sending, once for each mix on the way with each of the mixes public key. This encrypts the message in a layered fashion like a Russian Doll or an onion and is only possible because the sender chooses the path in advance and therefore know the public keys of the servers on the path. If a mix receives a batch of messages, it removes its own layer of encryption by decrypting the message, revealing only the next step and forwards the messages. This prevents a hostile mix from revealing sender and receiver of the messages. Since then, numerous systems have been designed (just to name a few which are not further mentioned in this thesis: I2P, Babel, Tarzan, Mixmaster, Crowds, Infranet, MorphMix, and many many more), as well as attacks on those systems [11].

The development has evolved more or less in two main directions: low and high latency anonymity systems.

2.3.1 Low Latency Systems

Low latency anonymity systems focus on interactivity, whereas the user requests something and gets an immediate response, like browsing the web, instant messaging or secure shell (SSH). Current systems can handle these examples of interaction already, but add a significant additional delay. Low latency interaction close to realtime, like anonymous voice over IP or other real time communication, are still not possible if a certain quality of services is needed.

A well designed and deployed anonymous low latency system is TOR (**T**he **O**nion **R**outer) and will be discussed in detail in the next chapter. Alternative systems will be presented shortly in section 3.2.

2.3.2 High Latency Systems

High latency anonymous systems on the other side focus on anonymity but without or only with very relaxed time constraints. These were the first deployed anonymity systems on the Internet, starting back in the early nineties. The exact time a message arrives is not as important as in low latency systems, although of course it has to be reasonable in context of the application and is usually between a few minutes and several hours. This is especially useful for one way communication like email, posting on the Usenet, weblogs or any other form of anonymous "publication". The big advantage is that messages get distributed and mixed over time, increasing the size of the anonymity set. For an adversary it is per se not distinguishable if a given arbitrary encrypted message is the one sent by Alice two hours or by Bob five minutes ago (assuming he knows somehow that both of these messages have been sent), making end-to-end confirmation attacks as well as certain other attacks much harder. There have been and are various deployed systems on the Internet, whereas the most popular are anonymous remailers which allow to send anonymous emails.

One of the first heavily used systems which started in 1992 was the pseudo-anonymous remailer anon.penet.fi by Johan Helsingius [38]. By sending an email to this server including the intended target, it was possible to send *pseudonymous* emails and postings to the Usenet. The remailer removed all the technical information which could have been used to identify the sender, and forwarded the message to the destination. To make replies and receiving of anonymous emails possible it maintained a database to map the pseudonymous ID to the real email address of every user, which made it not very secure as the server became a central trusted node. But in 1995 Helsingius was forced by the authorities to reveal the identity of a user because of a complaint filed by the Church of Scientology in the USA, although the server was located in Finland. Additionally, if the server was not reachable the service was not usable, and communication to and from the server was not encrypted which made it vulnerable for eavesdropping. The server was shut down by Helsingius in September 1996.

A deployed example at the time of writing this thesis is Mixminion [12] for anonymous remailing. It was first published in December 2002 and there are approximately 30 servers running [29]. It is a type III remailer and the successor of Mixmaster [31] and the Cypherpunk remailer [38]. The Cypherpunk remailer is a type I and Mixmaster a type II remailer. They all can be used (at least) to send anonymous emails, but Mixminion is the most advanced. The anon.penet.fi remailer was a type 0 remailer, as it only provided pseudonymity.

Cypherpunk The Cypherpunk remailer was the first well-deployed anonymous remailer system written by Eric Hughes and Hal Finney [38], starting in the early nineties. It is possible to use a chain of servers to hide the origin of a message, a technique used by many subsequent remailers. The messages themselves get encrypted with public key cryptography, and by distributing the trust on many servers there is an increased safety compared to a single anonymization server. Instead of compromising a single server, all the servers in the chain need to get compromised to defeat anonymity. By sending an email with the intended target and the predefined route of servers it is possible to send anonymous emails and postings to the Usenet, although it is not possible to answer to an email sent through Cypherpunk. The server, when receiving a message, decrypts it, replaces the senders address with the address of the server and forwards the message.

There is no built in prevention of replay attacks, which means that an adversary is able to resend a captured message at some time later. He does not need to know the content of the message, the encrypted message stays valid and the message would get retransmitted to the recipient. If the message for example was "Give the next person you see 50 Dollar", this would happen again if the receiver is not suspicious that he received the same message twice. Another weakness is that the mixes process the messages immediately, which makes the system vulnerable to statistical attacks and traffic analysis. Later on the "reply block" was added to the functionality, which made it possible to send emails to an anonymous recipient and to answer anonymous emails. It is a multiple times encrypted remailer message, containing a predefined route through the mix network with the creator of the reply block as final recipient. Due to the weaknesses and possible attacks it should not be used anymore.

Mixmaster Mixmaster was designed as successor of Cypherpunk to defeat traffic analysis by assuming that each network connection is compromised and a few mixes as well. This is a much harder threat model as in Cypherpunk, where only a few mixes were assumed to be compromised. Instead of processing the messages immediately each mix stores them and waits some time before sending out the messages in one big batch. Additionally, the messages size is padded so that each message has exactly the same size to make traffic analysis even harder. In Mixmaster it is not possible to answer an anonymous email or send mails to an anonymous recipient. To prevent replay attacks, each mix stores the message IDs of the processed messages, but the entries expire eventually so that the adversary only has to wait long enough until it becomes possible to resend a recorded message.

Mixminion Mixminion uses batch sending too, but every mix stores its messages in a message pool and chooses uniformly only a subset for each batch so each message is delayed for a random time. That is the reason why Mixminion is extremely resistant against traffic analysis. Additionally, all connections in Mixminion between the mixes are encrypted with TLS, making it even harder for an adversary. Replay of messages is not possible, as each mix stores a hash value of all processed messages and drops replays immediately. The cache entries do not expire, only if a mix changes its public and private keys all entries get dropped as the old messages are no longer decryptable anyway. This happens regularly, for example once a month. An additional feature of Mixminion is message reply, but more advanced then Cypherpunk: with so called "Single Use Reply Blocks" it is possible to answer the anonymous sender, so there was no more need for the insecure Cypherpunk network. For the intermediate mixes it is not distinguishable if the just processed message is a reply or not. On the current network between 80.000 and 100.000 messages get transmitted per day [28]. Similar to the TOR network the Mixminion network is not completely peer-to-peer based. The current network state is managed and distributed to the clients by a few central trusted and synchronized directory servers. If a new server joins the network it publishes its descriptor to the directory servers, who distribute this information to all the clients.

Nowadays, low latency anonymous systems might not seem so important anymore as high latency systems can be used to substitute some of their features. For example with anonymous email, it is possible to create a free email account with faked personal data at one of various freemail providers (for example gmx.de, gmail.com, yahoo.com or hotmail.com) absolute anonymously with TOR or JAP. These accounts can then be used to send and receive mails without leaking any information about yourself. This is not nearly as sophisticated and secure as anonymous remailers as it lacks (among other things) encryption and reordering of messages and provides only very weak anonymity, but it might be sufficient for some users. Care has to be taken in some jurisdictions as it might be illegal to provide fake personal informations to the service provider.

2.4 Attacks on Anonymity Systems

Attacks on anonymity systems always want to defeat anonymity, that is obvious. But this can happen in many different ways and with different goals in mind, depending on the power and initial position of an adversary, the assumed threat model for the anonymity system and the used technology. Linking a message with a sender or recipient without knowing the content of the message might be sufficient information in some threat scenarios to defeat relationship anonymity as well as getting to know the content of a message regardless of sender or recipient. All of the anonymity properties mentioned above like sender anonymity or unlinkability might be prone to subtle attacks. Although anonymous communication systems are a quite new area of research (less then 30 years), a vast number of different attacks was found, especially in the recent years. Due to this the following list of possible attacks is only a fraction and intended to give an overview of the different attack categories.

Cryptographic Attacks Many anonymity systems rely on cryptography in any way. If any of the used cryptographic protocols gets broken, the whole system might get useless. Even if many different protocols are used, which is state of the art in modern anonymity systems, the failure of only one of them might be enough. One example: in the original system of mixes by David Chaum [9], if as the only cryptosystem RSA was used it was shown by Birgit and Andreas Pfitzmann in 1990 that the mix system can be broken by an active attacker [42]. The attack was not new at that time, it was an already well known attack against RSA. It was just that apparently nobody thought about applying this attack to the Chaum mixes to defeat anonymity.

In May 2008 a weakness in the OpenSSL package of Debian was discovered [49]. The random number generator was predictable due to a programming error made in September 2006, rendering all keys generated since then with the help of the OpenSSL random number generator easily guessable. Instead of different sources of information to generate a lot of entropy, the only source of randomness was the program ID which is on Linux between 0 and

32767 (2^{15}). Many open source applications use OpenSSL for generating keys, a small list of examples: OpenSSH, OpenVPN, Apache2 and many mail transfer agents like exim4, Sendmail and postfix when using SSL/TLS, X.509 certificates, DNSSEC and TOR. TOR relies heavily on the OpenSSL package to generate the keys used in TOR, especially identity keys. This led to a whole bunch of possible attacks, even affecting clients not using Debian:

- At least three out of the six version 3 authority keys were known to be weak, which was very close to the majority vote needed for creating a network status consensus. For creating and getting clients into a complete fabricated TOR network, four out of six would have been needed, which was quite close. Still, if three authorities agree on a subset of the available TOR servers and claim to believe all other servers to be unusable, only the subset would have been marked as running since the other servers would not get enough votes.
- If all 3 nodes on a path were using weak keys, it was possible for an attacker sniffing the encrypted package from the local network or before the first relay to remove all layers of encryption, thus defeating anonymity.
- The attack above is unfortunately true for past traffic too. If somebody recorded TOR traffic in the past two years, it is possible that she can decrypt it.
- Hidden services use an .onion key to encrypt traffic with a hidden service to make it impossible to read or alter traffic. If the .onion key is weak, the client can not be sure anymore that he is talking to the original service as it might be impersonated.

Details of the TOR network can be found in section 3.1, whereas details on the used cryptographic protocols are discussed in section 3.1.2. Other possible attacks on the TOR network are presented in section 3.1.4.

Reputation Attacks The size of the anonymity set is vital for the possibly achievable degree of anonymity. Many systems like TOR or Mixminion also depend on volunteers who donate some bandwidth of their network connection and run a certain program in the background for the infrastructure of the anonymization network. Thus the reputation and the sustainability of the system are important factors besides the technical details like performance or used cryptography. Nobody wants to use or wants to help expanding a network that is used by terrorists, blackmailers or anyone else who seems socially unacceptable. If a system is always fighting not to get shutdown, it does not seem attractive to possible new users and it might be hard to keep the old users. An example: the anon.penet.fi remailer in section 2.3.2. Especially in bootstrapping a new system, the initial clientele and the perception of the social value can decide if it will work or if it will fail, regardless of the technical details and if it can provide only weak anonymity or real strong anonymity [15].

A sophisticated attacker might launch reputation attacks against the anonymization system, decreasing the perceived social value and the size of the anonymity set over time. It is not the kind of attack that delivers instant results, like breaking the encryption scheme or linking two communication partners together, but makes other attacks much easier.

Partition Attacks Splitting something into partitions based on any data can be a dangerous tool against anonymity systems, e.g. the users. Ideally, all users in the anonymity set are indistinguishable from each other, but in practice they are not. They in fact differ in many possible attributes and have sometimes very (maybe not unique but) special characteristics. Thus it is very important for an anonymity system to merge all possible users into the anonymity set as uniformly as possible. Possible attributes are:

- Time: The time a user is active might correlate with the timezone he lives in. Many users want to browse the Internet anonymously during the day and the evening, but probably not at 4 am.

- Language: If it is possible in the system that data from the user is transmitted in cleartext, this leaks information about the users language. This might provide little additional information about the user if the language is English, but might be valuable if it is e.g. Icelandic as the population of Iceland is about 320.000. If certain dialects are typical for a region, this could partition the users in chat rooms.

- Customization: In the Type I remailer it was possible to choose the cipher used for encrypting the messages as it used the OpenPGP libraries for symmetric and asymmetric ciphers. The user was confronted with the decision which cipher is the most secure, not only in a cryptographic way but also for anonymity. The weaker (cryptographic) cipher might be better for anonymity, as more people might use it which ends up in a bigger anonymity set. The regular user might not want to know if AES or Twofish is used for encryption, even possibly frightening the user with such possibilities and making him not to use the system at all. It was also possible to choose the size for message to get padded, but again this option might decrease anonymity instead of increasing it if an esoteric and very uncommon value is used. Details about this conflict can be found in [15].

- Message Pattern: Other message pattern like constant typos or use of only small letters can be used to further allocate users to different partitions respectively anonymity sets.

If there is no consensus and global view of the network it is also possible to partition the user on network level if an adversary tricks clients into believing bogus network states. If he can become an authority he might tell regular users a valid portion of the network and others a portion of the network he controls e.g. in TOR a spoofed directory with only servers and exit servers he controls. This would make traffic confirmation possible and defeating anonymity, by linking the victim with the messages content or communication partners with traffic analysis based on message time and volume pattern. In TOR this is prevented by signing the consensus from the majority of all directory servers, so the attacker needs to control more then half of the directory authorities for this kind of attack. It is not only applicable to TOR but any distributed anonymity system.

Passive Attacks Passive attacks are attacks where the attacker is able to watch the anonymity system, either from the inside or from the outside. Strong attackers can watch the whole network and control a large part of the network infrastructure and it is crucial for the design of every anonymity system if it wants to be protected against such strong attackers. The easiest passive attack is when the messages are transmitted unencrypted and an attacker controls parts of the infrastructure. Then he can read the messages that passes by without modifying them, just harvesting messages. In Mixminion TLS encrypted links protect against an passive adversary who can watch traffic between any two servers. The messages are encrypted themselves, but the additionally encrypted link makes it even harder.

Another powerful passive attack is “Traffic Analysis” and there is no easy protection against this in low latency anonymity systems [1, 10]. The timing and traffic volume entering and leaving the system have patterns and special characteristics which can be used to follow streams in the network and identify communication parties as everything that goes in should go come out again. Especially in times of low traffic this attack becomes easier. Without getting to know the content of the communication (which might be protected by encryption), the very fact that Alice is talking to Bob can be of value for an attacker. How traffic analysis can be used to attack the TOR network are discussed in section 3.1.4 (Traffic Analysis).

High latency systems are protected by adding large delays in message transportation, making end to end traffic confirmation very hard. In low latency anonymity systems this is difficult, as one of the desired features is interactivity and large delays are not feasible. They use instead a weaker threat model, which is that an attacker who watches both ends of a communication can correlate message timing and link the communicating parties. For the TOR network, this was demonstrated in 2005 by Steven Murdoch and George Danezis [33]. Details on this attack can be found in the section about attacks on the TOR network (3.1.4).
A possible attack to any low latency anonymity system might look like this:

1. Take anonymity system as a black box.
2. Try to get control over as many network links connecting the black box with the rest of the Internet.
3. Record everything that goes in and leaves the black box.
4. Analyze time and volume of messages and correlate communication parties.

Active Attacks An active attack is an attacks where the adversary is active, which means that he has at least to be part of the anonymity network. This is easy with many systems as they are open for new users, but difficult for systems like Freenet where it is possible to allow connections only to nodes that you personally trust (called "Darknet", `www.freenetproject.org`). Active attacks include pattern injection and modifications to the transmitted messages and creates some form of covert channel [8]. Some examples:

- Volume pattern: by sending the data in specially crafted bursts or packet sizes the stream becomes easily detectable, even if mixed with many other streams.
- Time pattern: sending at fixed or prearranged moments in time.
- Content pattern: changing some bits in the content might be undetectable for the employed security measures, e.g. in the TCP header or application data like graphical images.

Message or packet modifications include:

- Deleting a message
- Injecting a message
- Resending a previously recorded message
- Delaying a message
- Flipping some bits in encrypted messages. This makes either the whole message or the content from the first changed bit unreadable after decryption, depending on the used algorithm.

The system has to be designed to be as resistant as possible to active attacks, especially to already known ones.

Denial of Service Any public anonymity service is vulnerable to denial of service attacks of any kind. This can be from the inside of the system, or from the outside. An insider can take advantage of the system's design, operations like cryptographic ones that are computationally intensive or the system's internal logic, an outsider can use informations on the current state of the anonymity system to launch "classical" denial of service or distributed denial of service attacks. Some systems have special point of interests which would make sense to get attacked first, for example one of the servers in JAP: In the Java Anon Proxy, a cascade of mixes is used to hide the real address of the user. If any of the mixes in the mix cascade is temporarily not available because of a distributed denial of service attack, all the users who used this mix cascade have to choose a new one to continue surfing the web anonymously which of course increases the load on the other mixes. Details on JAP can be found in section 3.2.1.

Another example are the directory authorities in TOR which are central components in the TOR infrastructure. Although there are protections against it in TOR, any centralized topology is vulnerable at those central points. It is rather easy to block all communication with only a few computers on the Internet compared to a whole anonymity network with thousands of variable IP addresses for example at a firewall. Sometimes it is even feasible to block the website of an anonymity project to make it impossible for new users to download the needed binary files, which is also some kind of denial of service.

3 TOR & TOR Network Status

TOR (The Onion Router) is a well deployed overlay network that provides anonymity on the Internet for TCP based applications. This chapter explains in details the basic entities in TOR, the assembly of the TOR network and how the entities communicate and how TOR is being used today. It also shows possible attacks against the TOR network, open issues in TOR and alternative anonymity systems which are comparable to TOR. Finally TOR Network Status is presented, a project to view the current state of the TOR network.

3.1 The TOR Network

TOR [16] is the second generation of onion routing and is based on the original onion router [19] from 1999. It is an low latency anonymity system and used all over the world. The public network was deployed in October 2003 [17] and has grown from only a few servers to more then 2000 distinct servers with around 1200 servers running any time at the time of writing. It is an open source project and offers a lot of features for users seeking anonymous communication.

The Goal of the TOR network is to provide communication anonymity, thus making it hard for an attacker to link communication partners or link multiple conversations to or from a single user. It was developed for a high degree of deployability and usability to make it easy for people searching for anonymity to get it, resulting in an increased anonymity set. It is also easy for anyone to extend the underlying infrastructure by running a TOR node on their own with very low costs. It runs on a variety of platforms and is very flexible regarding what applications can be used and incorporates many good designs from previous anonymity systems.

Some of the features of TOR are:

- Easy application integration: every application which is able to use a SOCKS Proxy Server can be used with TOR. The original onion routing which failed in largescale deployment required a separate proxy for every application protocol, with most of them never written. Details on the communication inside the TOR network can be found in section 3.1.2.
- Location hidden services: This feature allows anyone to offer a TCP based service without revealing its IP address. Instead, all requests are

routed through the TOR network, thus making mutually anonymous entities possible. Although a special software is needed to access the service, no modification to the server and client applications is needed. These hidden services are propagated by special TOR nodes called “rendezvous points”, which when contacted will forward the request to the destination. An additional indirection requires the destination to accept the connection, to protect the service from a flood of requests and to respond to some request while ignoring others. This feature has been added to TOR in 2004.

- Variable exit policies: Every TOR server can be configured to prevent connections to special IP addresses or ports leaving the TOR network. This helps to prevent abuse of the TOR network as every TOR server operator can decide what and where he allows connections to the Internet from his server. Although it can be used to block applications based on the standard ports they use it is possible for software to miss-use those standard ports, e.g. Skype behind a restrictive firewall uses standard HTTP and HTTPS ports for connections, ports 80 & 443 [2]. By default outgoing emails are blocked when transmitted with SMTP (TCP port 25). Overall, this makes TOR more flexible and easier to deploy on a large scale then other anonymity services, for example if the server software runs on a corporate gateway this can prevent TOR clients to connect to the local network which would be a great security risk. When no exit traffic at all is allowed, the server becomes a so called middleman node and no traffic is transmitted to computers outside the TOR network.

- Variable bandwidth limits: Every TOR node operator can decide how much bandwidth to devote to the TOR network. This can be done either by bandwidth in kilobytes per second or in traffic volume for a given time period (e.g. 3 gigabytes per day, 10 gigabytes per month, and so on) and is especially useful if the Internet connection has a download limitation with additional data transfered beyond the limit generate additional costs.

- Open source: the source code is freely available. This is good for security as according to the Kerckhoffs principle everything about a cryptosystem should be public knowledge, except the key. It also increases portability, as it is possible for interested users to port the code to another platform. Other current anonymity systems like JAP or I2P are open source too, the Freedom Network code became available after the network was shut down. Right now TOR is available for Windows, MacOS X and various Linux/Unix flavors, but it should run on everything with a C compiler and some reasonable computing power for the AES crypto operations. The client software is also available as Java client called OnionCoffee, details on this can be found in the next chapter about client software.

- No special privileges needed: The installation of the client requires no special or superuser privileges, as there are no modifications done to the network stack or patches needed for the kernel. This is very good for usability as the software is much easier to install for the average user, and increases portability and deployability. For example the Freedom Network, an anonymity service developed and run by Zero Knowledge Inc. which is not longer operated since 2002, needed such privileges for installation [6].
- Forward secrecy: Different keys are used for encrypted communication, the TOR nodes negotiate short living session keys by using their longterm public and private keys. Once these short living keys are deleted, recorded communications can not get decrypted even if the node gets eventually compromised. This also has the advantage that symmetric cryptography can be used for communication, which is much faster then asymmetric cryptography.
- Full integrity checking: All traffic sent through a TOR circuit gets integrity checked, but only on the edges of a circuit (first and last node). This reduces the message overhead, prevents leakage of information about the position of the node in the stream and protects against malicious nodes that, although they can not read the communication content because of encryption, still can change the ciphertext. The resulting plaintext would make no sense and this would get detected by the integrity check, thus the circuit would immediately get teared down. The earlier design of onion routing used a stream cipher without any integrity checking.

The threat model of TOR is somehow weaker as in other anonymity systems. The most commonly assumed threat is a global passive adversary who can monitor all traffic going into and out of an anonymity systems. Additionally he is often assumed to be able to inject, delete or modify messages in transit, and a subset of the nodes can be under his control. It is unrealistic that he controls all of the nodes. Anonymity systems that protect against such attackers can be assumed to be secure for most of the possible users, one example for such a strong anonymity system is JAP. TOR by design does not protect against such a strong adversary. By watching Alice and Bob and analyzing the time and volume pattern the adversary can and eventually will find out that they are communicating. For this attack a suspicion is required that two specific parties are communicating. The adversary assumed in TOR is able to run or control a subset of network nodes, is able to observe some fraction of the network traffic and is able to inject, delete and modify messages [16].

TOR protects against traffic analysis attacks which would allow him to find out which nodes of importance to attack only by watching traffic pattern. In favor of a simple and deployable design, some commonly used techniques in anonymity like message reordering were intentionally not used when TOR was designed. The network is not solely peer-to-peer based and is not using stenography, thus it is not hiding the fact that somebody uses TOR. It would be in fact quite easy for an adversary to find out if somebody specific uses TOR, just by watching the network traffic e.g. by comparing the IP destination address of every packet with the current set of TOR servers. Another design choice was to keep TOR completely separated from protocol normalization: many protocols like HTTP are very complex and variable, which would require complex protocol normalization to make all clients look the same. For HTTP TOR relies on Privoxy, a filtering proxy software which removes everything that could leak identity, for other protocols this has to be considered separately before relying on the anonymity provided by TOR.

As TOR is a low latency anonymity system, techniques like traffic mixing and link padding were abandoned as they might significantly harm response times and experienced interactivity by the user. One of the last points which might be considered a drawback of TOR is the lack of UDP transportation. Many applications like multimedia streams, DNS or Voice over IP use UDP instead of TCP for transportation, as it is lightweight and faster then TCP: these applications must either be tunneled over TCP before using TOR rendering all the benefits of UDP over TCP useless, or must use another software for anonymity.

All this design choices made TOR maybe not the best solution regarding the technical details, but increased the size of the anonymity set and the number of users significantly. This somehow paradox relation was in detail considered by the creators of TOR [15].

3.1.1 Basic entities

The TOR network consists of several basic entities, servers and client software. The different kind of servers form the infrastructure and are needed for providing anonymity to the clients. All TOR software require no special privileges so it is possible to run it even in restricted environments. The very details on TOR can be found in the design paper [16].

Directory Server The core element of the TOR network are the directory servers, also known as authorities. As the TOR network is constantly changing and new servers can join the network any time every client needs to know the current set of servers. This information as well as the servers state and exit policies is called “directory” and is distributed by the trusted directory servers via HTTP. This approach is different to anonymity systems which are completely decentralized as they need to distribute this knowledge in-band. The drawback of the in-band method is that due to network delays (intentionally or not) it can take some time till all the clients are aware of the new network state. An adversary might exploit the fact that some clients have a different view of the network compared to others for their advantage, even if it is only for a short period of time. All directory servers are synchronized and redundant, and need to agree on a common directory by majority voting. This prevents an adversary to set up a malicious directory server or take over control of an existing one. This means that an attacker needs control over more then half of the directory servers to tell different network states to the users, for example only TOR nodes he is in control of, which would completely defeat anonymity.
At the time of writing there are eight directory servers, located in the United States of America and Europe (two in Germany, two in the Netherlands and one in Austria). When the network was deployed in 2003, only three directory servers were used. To limit the network load on the directory servers, all TOR servers distribute the signed consensus. This is no security risk, as the public keys of the directory servers are included in the source code and it can be verified that the directory is not malicious.

Server Most of the TOR servers are operated by volunteers, anyone can download the server software and run a TOR server. As soon as the server program is started it publishes its keys and descriptor to the directory servers, and will then become part of the directory. Once the server is listed in the directory it can be used by clients. It will continue to publish signed statements to the directory servers constantly. To be as flexible as possible, every server operator can decide to limit the bandwidth or transfered data TOR might consume, and whether the server will be an exit server and allow connections to hosts outside the TOR network. There are many other configurable settings.

All servers listed in the directory have certain flags set, according to their capabilities. Important flags are:

- "Running": Whether the server is running right now, or not. This is important for the clients path selection and is set if the authority was able to connect to it in the last 30 minutes.
- "Hibernating": If the server is running, but not ready to allow circuits. This happens when the data transfer limit was reached, and the server will hibernate until some time in the next accounting period.
- "Exit": A server is labeled as exit server if it is an exit server allowing connections to certain important ports like 80 or 6667.
- "Bad Exit": If the server is believed or suspected to do malicious things like man in the middle attacks, this flag tells the clients not to use it for exit connections. This flag is set by the TOR maintainers, by hand, often after discussions on the TOR mailing list. In the future this might get automated by e.g. TORFlow which tries to verify exit nodes and that they do not alter or redirect the traffic.
- "Fast": If the server is running on a high speed Internet connection with at least 100 kilobytes per second it is flagged as a fast server.
- "Valid": A server is valid if it is running a version of TOR not known to be broken.

There are many reasons why someone would run a TOR server: it might be that the operator has a personal interest in privacy issues, has a spare broadband Internet connection or wants to get a certain degree of deniability if he runs an exit server. It might also be an organization that uses TOR and would like to use their own TOR server as first or last server, to make traffic analysis harder or because of any other reason.
The distribution of countries where TOR servers are operated is interesting, as by far the vast majority is operated in the United States of America and Germany. This distribution and its consequences is further investigated in chapter 5.

Clients The client software runs on the users computer. The specification is publicly available, so anyone could decide to write their own client with maybe additional features. Right now there are two software clients available, the original client written in C and available for the majority of popular operating systems (Windows, Mac OS X and various Linux/Unix flavors) and OnionCoffee, written in Java by the RWTH Aachen and therefor usable on any platform with a Java runtime environment.
Every client needs a complete picture of the current TOR network which is distributed by the directory servers. Once a sufficient part of network informations is downloaded after bootstrapping, the client software takes care of opening and closing the circuits, some preemptive and some on demand. Details on circuits, path selection and the involved cryptography can be found in section 3.1.2. The client software can be configured in many ways, for example to avoid a given list of servers or forcing it to use only a given subset of servers.
It is also important that the core TOR client is not enough for all situations to provide anonymity: many complex and variable protocols like HTTP need extra care to prevent them from leaking information. Especially tracking cookies and interactive code shipment with code snippets getting executed on the clients machine, like JavaScript or Java applets in HTML pages, can be used by a malicious website to identify the user even if the originating IP address seems to change constantly. This has somehow similar been studied with other anonymity systems e.g. SafeWeb, an anonymity web proxy system [24] in 2002. That is the reason why the TOR client gets shipped in a bundle with Privoxy, an additional privacy enhancing web proxy, Vidalia and other software.

Guard Nodes One of the problems with path selection is that if the client software chooses all 3 nodes in a circuit at random an attacker will eventually control the entry and the exit node. To prevent this, guard nodes have been implemented which will be used as entry nodes to reduce the probability of selecting a malicious entry node for any circuit. Each client chooses a list of guard nodes when starting the first time and saves them persistently on disk in a fixed order. For every circuit TOR chooses at random the guard nodes in the list to use. If less then two guard nodes in the list are currently useable, the list gets extended with new nodes but without replacing the old ones. Guard Nodes were not part of the original design but were added later because of a newly discovered attack [35] which made it to identify the IP address of a hidden server (see "Locating Hidden Servers" in chapter 3.1.4 for details).

Bridge Relays Bridge relays are just like normal Tor servers except they don't publish their descriptors to the main directory authority, instead they publish it to the bridge authority. The only difference between the normal directory and the bridge directory is that the bridge directory is not publicly available. It is possible to get the current set of TOR servers, but difficult to get the current set of bridges. This was not part of the initial design, but was added later to increase censorship resistance [14]. Somebody who wants to stop other using an anonymity system like TOR only need to block access to the current set of TOR servers. To bypass this, users need to know only one bridge which is not in the current set of TOR servers.
Clients get to know bridges differently then they get to know the regular TOR servers: They can request information on bridges from the bridge authorities, by email, by a special website or ask their friends for a bridge and will always get only less then a handful of IP addresses. One of these bridges will then be used for all further connections to the TOR network, bypassing every filter to the regular TOR network. This implies that only servers with a high availability are useful as bridges.

3.1.2 Communication

The design of onion routing is not new. First introduced in 1981 by Chaum [9] it was subsequently improved [19] to the current state, TOR [16]. TOR is an overlay network. This chapter will explain how the communication in the TOR network works.

Communication Units Communication in TOR works on the transport layer, anonymizing TCP streams. This means that a TCP connection chooses once a circuit and sticks to it as long as the circuit or the TCP stream exists. Traffic on those circuits is sent in fixed size cells to make traffic analysis harder, but without any reordering or artificial delays.

- Stream: Instead of IP packets or TCP packets, TOR works with TCP streams. This has the advantage that the experienced performance for the user is better compared to TCP packet anonymization. Packet anonymization was proposed in [23] and has the drawback of possibly causing a lot of resend requests as any packets can take completely different paths with arbitrary delays to the destination. On the other hand it is able to use the lightweight UDP protocol and is protected against certain attacks. Switching to IP transportation or using UDP as underlying protocol is a never ending discussion in the TOR community.

- Circuit: A circuit is the path of TOR servers used for communication. Many TCP streams can share a circuit, and by default a circuit consists of three TOR servers: the *entrance node*, the *middleman node* and the *exit node*. The entrance node is aware of the real IP address of the users, whereas the exit node is able to see the content the user was transmitting and the final destination. The circuit is constructed incrementally by the client software. It is possible to create paths with only two servers, or with up to an arbitrary number of middleman nodes. The client software also constructs some circuits preemptively and adapts the current number of circuits to the usage pattern of the user.

- Cells: The data units passed along the circuit are called cells and are encrypted/decrypted at every node in the circuit. They have a fixed size of 512 bytes and consist of a header and a payload. There are two different kind of cells, control cells and relay cells. Control cells are used for intra-TOR communication like key exchanges or circuit teardown, while relay cells are supposed to leave the TOR network at the exit server.

With this communication units the TOR network is based on loosely federated clients and servers without much communication overhead.

Cryptography TOR relies heavily on various cryptographic protocols:

- TLS: All TOR routers use TLS to encrypt communication with other TOR routers. This provides perfect forward secrecy and prevents an attacker from modifying data or impersonating a TOR router.

- Asymmetric cryptography: Every TOR router uses a long-term identity key and a short-term onion key. The first is used to sign the TLS certificates and the routers descriptor for the directory servers, the second is used for ephemeral key negotiation with the clients and for encryption and decryption of circuit communication. Another public key is used for identifying hidden services.

- Symmetric cryptography: All cells are encrypted using 128 bit AES.

- Diffie Hellman key exchange: The short-living symmetric keys are negotiated between the client and the TOR nodes on a circuit incrementally with the Diffie Hellman key exchange.

- Hash function: The SHA-1 digest is used to check data integrity at the edges of the each stream to prevent an attacker from modifying the encrypted data without the client or the server to notice.

Stronger cryptographic protocols would have been available (e.g. 256 bit AES), but were abandoned due to the increased required performance and because the main goal of encryption in TOR is to provide unlinkability. Still, the used encryption protocols provide a good tradeoff between security and performance. If the user needs confidentiality he has to use an additional layer of encryption (e.g. TLS or SSL) on top of TOR. Everything leaving the TOR network is otherwise transmitted in plaintext, even visible to the TOR exit server.

Path Selection The algorithm how TOR chooses a path of servers to open circuits has been widely studied in the past. It is the most crucial part in weighting the compromise between anonymity and performance, so studies have been made to optimize either performance [37, 45, 36], security [3, 35] or both [46, 34]. Details on the current implementation in TOR is documented in the TOR path selection specification [51].

3.1.3 Usage Pattern

As TOR provides anonymity it is hard to tell how many people use it. It is well deployed, and it is expected to have tens of thousands of users. Researches studied the way TOR is being used in the beginning of the year 2007 [26] and the beginning of the year 2008 [25] by running a high bandwidth TOR server at their university. In 2008 they ran an exit server at first for a few days to collect plaintext application data headers to analyze how TOR is being used. Simply analyzing the destination port numbers would have yielded incorrect results, as applications which use non-standard ports like Skype would have biased the distribution. Afterwards they used the same server for about 2 weeks as entrance server by refusing any exit traffic to study the users of TOR, and where they come from by analyzing their IP address. The analysis from 2007 was slightly different.
The vast majority of TCP connections made through TOR was caused by HTTP traffic whereas a disproportionately big part of the transfered data was caused by BitTorrent, a common peer-to-peer filesharing protocol. This indicates that many people use TOR in the way it was originally planned and designed for, by using interactive protocols like HTTP, SSH and instant messaging. On the other hand, TOR is commonly misused for various

kind of disputable actions and attacks on computer systems, e.g. websites defacements or downloading large files with the BitTorrent protocol, thus hurting the overall performance and consuming a lot of bandwidth of the TOR network. As a result, many exit node operators were and still are getting contacted by the police and suspected to be the source of action.
Other interesting characteristics of the TOR network are that most of the users come from Germany, the United States of America and China and during the two weeks in 2008 as entrance server connections from 126 different countries were observed. This was similar in 2007. Some of the other protocols used included insecure protocols like POP, IMAP, Telnet and FTP. As the encrypted chain of TOR servers ends at the exit server, misbehaving exit servers can sniff the traffic leaving their server for the final destination. This is especially dangerous as those insecure protocols transport everything (even passwords) in plaintext. Additionally, these insecure connections can share the same circuit with secure protocols, thus not only revealing login names and passwords to the exit server operator but also defeating the anonymity of the secure connections. A possible way of protecting the users from leaking their passwords would be to block those insecure ports by default, but this has not been incorporated into TOR and might never happen as the TOR maintainers want to keep it as flexible and open as possible.

3.1.4 Possible Attacks against TOR

Several attacks have been discovered as TOR became more popular in recent years. As a rough categorization, attacks can happen either inside of the TOR network, or from the outside. They can be done either actively like man in the middle attacks or message modification, or passively like message eavesdropping. Passive attacks are much harder to detect but they are preventable. Some of the recent attacks are presented here, while a complete list is of all possible attacks is not possible.

The "Embassy Hack" A recent attack which probably drew the most attention of the media to the TOR network ever was the so called "Embassy Hack" in 2007. A Swedish hacker named Dan Egerstad published in his weblog a list of 100 sensitive email account informations, including passwords. These email accounts belonged to embassies from countries all over the world, although other sensitive information and many other email logins have been captured but not published, including private persons and company accounts. They were collected by monitoring a bunch of TOR exit servers, logging all unencrypted traffic from the TOR network into the Internet and searching for passwords.

This drew a lot of attention to the TOR network, many users seemed and probably still seem to mis-use TOR. They apparently relied on TOR for security, not anonymity. But as TOR only encrypts traffic within the TOR network, all the traffic is observable to the exit server in plain text if no additional security or encryption is used (like SSL, TLS or HTTPS). This is obvious to the technical advanced users, but can not be expected for the average user. The embassies email informations have been chosen by Egerstad to draw attention to the problem, which succeeded. Details on the problem of malicious exit servers can be found in section 3.1.5 (Traffic and Exit Server Verification).

Sybil Attack Almost all anonymity systems that rely on distributed trust are vulnerable to the Sybil attack [18]. This is true for TOR as well as almost all other anonymity systems. It is based on the idea that a small number of entities can counterfeit multiple identities. In fact, all distributed systems without a central authority are vulnerable to this attack. Given enough resources an adversary can introduce a large number of malicious nodes which can compromise an entire distributed system. Public key cryptography, as it is used in many distributed anonymity systems, is only feasible to verify if someone is the one he claims to be. It is not usable against an adversary forging many identities and adding an enormous amount of malicious nodes. Only a central authority can protect against this attack, which of course is contrary and hard to deploy in an anonymity system. TOR in particular only protects against an adversary if he is unable to control the first and the last node in a circuit. By controlling enough TOR servers, the probability increases that eventually the adversary is able to link communication partners and to defeat anonymity. This has led to the introduction of guard nodes (see section 3.1.1).

Traffic Analysis Traffic analysis is a young field in computer science [1, 10]. It uses metadata from network communications instead of content to attack anonymity systems, as content is encrypted with strong ciphers. This metadata includes volume and timing information as well as source and destination of messages. TOR does not protect against a global passive adversary that can monitor all network connections and thus trivially follow messages in every circuit due to special patterns. But traffic analysis makes it possible to link communication partners at random, just by controlling and monitoring some part of the network. This has been shown in [33] by estimating the traffic load on specific TOR nodes. All the circuits going through a TOR node share resources like CPU or network connection, and all the circuits are served in a round robin fashion. By adding a circuit the load on the node increases, which affects and increases the latency on all other circuits running through that node. This fact can be used to determine if a certain circuit runs over a specific node. In combination with a malicious destination server that sends traffic in a special pattern it becomes possible to identify all nodes in a users circuit. It is even possible to defeat anonymity if additionally the entry server belongs to the adversary. The costs for identifying the nodes in a circuit in such a way are fairly low, but quite high if it is also used to defeat anonymity as numerous entrance servers are required.

Locating Hidden Servers In the original design it was possible to locate a server offering a hidden service because of a flaw in the design of the path selection algorithm. Øvelier and Syverson [35] were able to find out the IP address of the hidden server within a few hours by running a single TOR middleman node and a client connecting to the hidden service, with only slightly modified software. The attack was much faster if the hidden server was running a TOR server as well, decreasing the attack time to only a few minutes. They used a special time and volume pattern in their clients communication and created a lot of circuits, waiting for the hidden server to use their middleman node as the entry node for the circuit to the rendezvous point. If their middleman node was the first node, the pattern as well as the IP address of the hidden server were observable to the middleman node. This attack was the reason for the adoption of guard nodes, where the clients respectively the hidden services use a relatively small fixed set of entry nodes.

In another attack by Murdoch [32] it was shown that it is possible to verify if a given IP address is running a certain hidden service, something the hidden service is supposed to prevent by design. The attack is based on the imperfectness of hardware and the fact that processes on the same computer share the same resources. According to the temperature of the crystal oscillator and inside the chassis the computer's clock detectably varies over time. By imposing heavy load on the CPU the temperature inside the computer rises, which changes the clock skew and is remotely detectable. This can be measured remotely either by TCP timestamps, ICMP timestamp requests, on some operation systems from the TCP sequence number or with timestamps on the application layer. This can be done with a unique pattern, and searching for that pattern over time can be used to identify the host running the hidden service, and thus defeating anonymity. In the experiment conducted by Murdoch it was sufficient to increase the servers temperature by 1.5° Celsius. The downside of this attack is that it can become costly if all the servers have to be tested.

Debian Weak Keys TOR was affected by a weakness in the OpenSSL random number generator. For details on possible attacks based on the Debian OpenSSL flaw, see section 2.4 (Cryptographic Attacks).

Website Fingerprinting TOR is vulnerable to the so called website fingerprinting attack which is a subset of traffic analysis [21]. The communication pattern of website browsing is easily distinguishable from other traffic, and every website has quite unique traffic pattern because of the timing and volume of transfered data, e.g. affected by the number of images and the image sizes. It is possible to create fingerprints of the most popular or most interesting websites by saving these pattern. Afterwards it becomes possible for an entry server to compare the users traffic pattern with any of the website fingerprints. This makes is possible to tell if the user visited a given website or if he did it likely or less likely, thus defeating anonymity.
This attack is also applicable to other interactive protocols and almost any low latency anonymity system if they don't delay or reorder packets. One famous example for another protocol is the timing attack against SSH from 2001 [47].

3.1.5 Open Issues with TOR

The TOR development team still works on TOR and improves it, at the same time TOR is studied heavily in the academic area. Some of the ongoing discussions which will eventually become a part of TOR are presented in this chapter.

Link Padding As TOR does not by design defend against traffic analysis there exists a long term discussion about link padding and if it should be included into TOR and how. Link padding would defeat traffic analysis. Two exemplary methods of link padding would be feasible with a third one which was proposed recently:

- Constant link padding: The link between two communication parties is filled with a constant rate of packets. Between the constant rate of dummy packets the real communication packets get inserted.
- Random link padding: The packets are sent with a predefined random schedule.

- Dependent link padding [55]: This approach is new and promising in low latency systems, as constraints on the delay are possible. The loss of performance compared to the overhead is relatively small and might vary if various distribution models were implemented.

If any of those approaches will become part of TOR is unknown, as all of them decrease performance respectively increase the overhead considerably.

Windows TOR Server Non-server versions of Microsoft Windows have sometimes a problem running a TOR server, especially systems with low memory. After some time the TOR server crashes with an "WSAENOBUFS error" because the available buffer space gets exhausted. TOR uses the so called non-page buffer pool instead of the native Windows networking system calls, which is limited in size. Only a reboot can fix the overloaded TCP/IP stack, affected versions of Windows are Microsoft Windows 98, ME, 2000, and XP.
There seems to be no real fix for this, except adding more memory or limiting the TOR data rate. Running TOR inside of a virtual machine would be an alternative workaround, e.g. JanusVM or the Incognito live CD.

Traffic and Exit Server Verification Another field of active research and development is traffic and exit node verification. As unencrypted traffic passes the exit server it is possible that a malicious server alters the communication content. This would hardly be detectable to the user. A malicious exit server might as well forge the destinations certificate during connection setup and might conduct a man in the middle attack, for example by offering his own SSL certificate. In this case the user would get an error message that the certificate has changed, but this might get ignored by the average user.

One approach to counter this was a Google Summer of Code project last year called TORFlow and is still under development. TORFlow is a set of python scripts and tries to verify exit nodes with test data by making circuits through specified exit servers and comparing the anonymized communication content with the expected content. For the user this is also possible, as he can specify the exit server or servers to use, e.g. in his configuration file, but not always visible as the exit server might inject for example an invisible IFRAME to load malware onto the users computer. It is also possible to specify the exit server within the requested URL: `hostname.TORfingerprint.exit`, which is sometimes handy if some websites are restricted to specific countries or IP ranges. Coming back to

TORFlow, by comparing the expected content with the actual content it becomes possible to detect malicious exit nodes and label them as invalid, thus removing them from the network consensus.

A more difficult problem is to detect whether an exit server is sniffing on the communication content. It might be obvious if the server has a obviously evil exit policy like allowing only unencrypted protocols. This would permit the server operator to log unencrypted passwords as it was shown in the "Embassy Hack“ in section 3.1.4, but would be obvious to advanced users that know how to check their exit servers exit policy. Frequently used protocols that transmit passwords in cleartext are POP, IMAP, FTP and telnet. To prevent discovery an adversary would have to cloak this by allowing secure protocols.

Protocol Changes The protocols used inside of TOR are under constant evolution. One example which might get changed in the near future is the bootstrapping mechanism, which happens when the client software connects the first time to the network. It first downloads all server descriptors, the servers public keys, their exit policies and more. This knowledge is at the moment around 1-2 megabytes in size, thus it takes already very long for dial up or other slow Internet connections. Another open point is the distribution of the consensus, which at the moment is downloaded as a whole. In the future it might make sense to download only the differences between two consecutive consensus, which would decrease the data required to download periodically considerably.

3.2 Alternatives to TOR

With the growth of the Internet, numerous solutions have been realized to provide anonymity. They differ widely in their technical details, depending on the target audience and the layer of communication they work on. Some of them try to provide only application specific protocols, like HTTP for browsing the web, others try to offer as much flexibility to the user as possible, e.g. by anonymizing IP packets. A few alternatives to TOR are described briefly in this chapter and are compared with TOR. Probably the oldest company active in this field is Anonymizer Inc. (`anonymizer.com`), founded in 1995 by one of the authors of the Mixmaster remailer software.

3.2.1 JAP a.k.a JonDo

The Java Anon Proxy [4, 5], nowadays known as "JonDo", is the most similar, well deployed anonymity software in comparison to TOR and was first published in 2000. It was part of the research project "AN.ON" by the University of Dresden and the University of Regensburg together with the privacy commissioner of Schleswig-Holstein. It is very similar to the earlier design of ISDN mixes [41]. When the project ran out, a spin off company named JonDonym continued to develop and maintain the software, renaming it from JAP to JonDo. JAP used to be free of charge, but with the evolution into JonDo the company started to offer a prepaid version with high bandwidth and many other features like servers in different jurisdictions. There is still a free version that can be used, but it only works for HTTP with no guaranteed bandwidth. The prepaid version guarantees at least 64 kbit per second for every user which is as fast as single channel ISDN, and can be used to connect to all Internet ports (like SSH and FTP).

Technical Details JonDo aka JAP is written in Java, so it is usable on all operating systems that have a Java runtime environment. The server software is written in C++ for higher performance [22]. As in TOR, it uses symmetric and asymmetric cryptography for encryption. The communication happens through so called MIX cascades, whereas a cascade is made of at least two MIX servers, by default three servers are used. It is fixed in order, and the anonymity set for a user is the number of users that use the cascade concurrently. TOR in contrast chooses three servers at random, periodically and constantly. The current number of users that use a cascade is provided as information to the users, as fewer users mean higher bandwidth but a smaller anonymity set and vice versa.

Inside the MIX cascade communication happens in fixed size slices. They are encrypted at the client in a layered fashion, as in TOR. But in contrast they get pooled and transmitted by the MIXes in rounds, with injected dummy traffic in both directions if a client has currently no data to send or receive. The throughput of a channel is adaptive to the current bandwidth needs of the user. To defeat traffic analysis the slices are reordered inside the MIXes. Due to the reordering it is not necessary to hide where the slices enter and leave the anonymity network. Still, the delay observable at the client is suitable for interactive applications like web browsing or remote shell access.

The threat model is stronger then the one used in TOR, as only one honest MIX server is needed to provide anonymity (remember, TOR needs two honest servers, namely the first and the last for every circuit). All communication links and all MIX servers except one in the current cascade can be controlled by the adversary, still it is not possible for him to link communication partners. To prevent flooding inside the cascade and to allow the use of the payment service the client software authenticates itself to the mix servers, but without user identification.

3.2.2 Open Proxies

There exist a long list of Proxy servers that are publicly reachable over the Internet, e.g. at `xroxy.com`. Some of them might serve the purpose to attract users seeking anonymity, but most of them might simply be misconfigured. The user has to choose one of them manually which makes their usage very complicated compared to other services and these proxies can vary a lot in quality, namely usable bandwidth and reachability. Many of them do not use encryption at all, so if the adversary is wiretapping the user there is no increased security. There also exists various specialized proxy servers software packages:

Psiphon Psiphon is a special web proxy software available for free and designed to circumvent Internet censorship. It relies on small trust networks, users in restrictive countries connect to a computer they know and trust outside of their country with a username and password. They are able to browse the Internet without any other software then the browser, the traffic is encrypted with HTTPS. It provides no anonymity, only censorship circumvention.

CGIProxy CGI Proxy is similar to Psiphon, except that it is not an independent software but a CGI program written in Perl and reachable via a web server. Various websites with lists of open CGI Proxies exist, but again they provide no anonymity because of the log files at the proxy server and lack of encryption. They can only be used for web traffic and to bypass content filtering. Something similar to CGI Proxies are PHProxies where the program is a PHP script, but with roughly the same functionality.

Compared to TOR, open proxies provide far less security and privacy. The user has to trust a single proxy server, which might be quite controlled by an adversary. Compared to the real world this would be like giving your postcards and private mails to a complete stranger, and asking him to take it to the post office for you.

3.2.3 Commercial Proxies

There are many companies that offer proxy or VPN service for anonymous Internet access, some charge a monthly fee. Most of them use only one computer system to hide the originator, no cascades are used. Depending on the company, either full VPN connections can be used, or proxy servers. The problem with those services is that the user has to fully trust a single company, which might be located somewhere in the world. An exemplary selection: `anonymizer.com`, `blacklogic.com`, `aliveproy.com`, `vpnprivacy.com` and many, many more. At least in some jurisdictions authorities would easily have access to the user data, defeating anonymity would be straight forward. The benefit of such systems on the other hand is, that they are quite easy to deploy and can be faster then distributed trust systems. Customers can count on that a certain bandwidth is available for them, or some other possible kind of "quality of service" value like reachability. The only thing needed for running such a service are some servers in a data center with a high bandwidth Internet connection, and some software. If for the users performance is much more important then real (or any) privacy, these services might be worth a thought.

3.3 TOR Status

TOR Status, also known as TOR Network Status, is one of many open source projects around the TOR Network [52]. Its goal is to present the current status of the TOR network to the interested user. In essence it consists of a Perl script which connects to a local and thus required TOR server and writes the necessary information into a database. On top of this a PHP application takes care of the appropriate presentation. TOR Status is designed to run under Linux but might run under Windows as well, although there are still open problems running a TOR server under Windows. For details on the problem running a TOR server under Windows, see 3.1.5 (Windows TOR Server).

Frontend The frontend is a website written in PHP. It connects to the TOR Status database and presents details on all the current TOR servers as well as summarized values of interest. For every server in the current network details like uptime or transfered bandwidth are presented. It is possible to search the servers for certain attributes, or check if a given IP is an exit server. The website is hosted on numerous mirrors and is also accessible as TOR hidden service.

Backend The backend is a Perl script in combination with a database. The Perl script connects to a local TOR control port to get the current state of the network periodically, thus to host a copy of the TOR network status website a local TOR server is required. It parses the output, writes the needed informations into the database and generates the statistical charts. Almost any database can be used, the only requirement is that a Perl module for the database connections is available.

It is important to notice that only the **current** state of the network is viewable, there are no informations saved in TOR Status how the network changed over time. This will be addressed in the next chapters, which is the practical part of this master thesis.

4 Monitoring the TOR network

The previous chapters were intended to give the reader an overview over existing anonymity services and how they work. This chapter is about the TOR network, and how monitoring the TOR networks infrastructure can improve overall anonymity. The first part discusses the values of interest that are worth monitoring and how they can be used to improve the degree of anonymity. As a next step these values are evaluated if they could possibly decrease anonymity the degree of anonymity. Furthermore the practical data collection and how it was implemented is described.

4.1 Values of Interest

Before monitoring a system it is important to identify the values of interest, i.e. not all values are bearing useful information. As with the TOR network it is important to find values that have a direct influence on the users anonymity, but without introducing possibilities for new attacks on the network, without decreasing the degree of anonymity and making existing attacks easier. The TOR infrastructure is a dynamic number of TOR servers operated by volunteers, so the values of interest should focus on the TOR servers.

Exemplary values that would harm anonymity if collected globally are:

- Number of users: This would make the absolute size of the anonymity set easily retrievable for an adversary. If this value is unknown the probability that a given circuit belongs to a certain user is much harder to calculate.
- Number of connections a TOR server handles: .it becomes easier to calculate certain probabilities if for every server the number of concurrently handled streams is known. This has been used in [33] to make traffic analysis remotely possible, respectively much easier.
- Circuit throughput: Time and volume pattern in user traffic should **not at all** get collected globally, as this would easily allow various attacks based on traffic analysis.

Monitoring the infrastructure of the TOR network has been done before but somehow was suspended again, or the data is at the moment not available to the public. There is a website that collected the number of TOR nodes and the total traffic of the TOR network over time, [53], but stopped collecting those values as of August 2007. Instead it refers to the current TOR Status websites, which only show the current state of the network and do not monitor anything over time (yet). One of the design papers by the creators of TOR [17] published some statistics about the number of running routers and the total relayed traffic in the early stages of the TOR network, from August 2004 till January 2005. Apart from that no official statistics are available.

While working on the subject of TOR infrastructure monitoring and inspired by those early data collections the following important values worth monitoring have been identified:

- Total number of running servers
- Total number of running exit servers
- Total number of running fast servers
- Total number of running guard servers
- Country distribution of running servers
- Country distribution of running exit servers

These values have been chosen with two goals in mind: to measure the effects of the infrastructure on the possible degree of anonymity and on the other hand to measure the effects on the quality of service. Bad anonymity per se does not attract further users, as well as bad performance.

4.1.1 Total numbers

For the overall anonymity and security, the number of TOR servers at the time of usage is crucial. A handful of servers would be easy to attack, either directly with the goal of defeating anonymity or with a denial of service attack. Thus the greater the number of TOR servers, the better. This is also true for Hidden services. They have not been discussed in detail, but they provide additional resistance to denial of service attacks compared to traditional web services because of their design and integration in the TOR network.

In the network consensus all servers are flagged according to their capabilities and qualities. The important flags for the degree of anonymity are "running" and "running" in combination with "exit server". Only those are the servers a client uses for his circuits, as apparently a server that is not reachable is of no value in a TOR circuit. Additional server values influence the clients path selection, but the path selection algorithm in TOR might change in the future due to newly discovered attacks or improved versions. Thus the number of running servers and running exit servers is the smallest possible metric with influence on the path selection and anonymity.

For the quality of the TOR connections, two other flags become important, namely the "fast" and "guard" nodes: the more fast TOR nodes there are, the better becomes the overall performance for the client. Of course, every TOR circuit can only be as fast as its slowest TOR server, but the more fast nodes in the network the higher the probability that all nodes will be fast nodes and provide enough bandwidth. Especially asynchronous communications like web browsing or multimedia streams, where short requests can initiate large downloads, transmit data disproportional in one direction and need high bandwidths. Another example would be a large upload from the client, like picture publishing or video file uploading. Thus the number of fast nodes is an important metric for the quality of the TOR network. The guard nodes, on the other hand, are important for the quality of service as well as the provided anonymity. The more guard nodes in the network, the better the performance (as the proportion guard nodes to clients decreases) and the number of clients a given guard handles decreases. For the anonymity the number of guard nodes is important as the probability of guard nodes to malicious guard nodes decreases, assuming that most of the guard nodes are operated by honest operators. This of course does not hold against Sybil attacks.

All the other server flags like "named", "hibernating", "stable" or "authority" are of not such importance, neither for the anonymity nor the quality of service. The number of authority servers is unlikely to change often, as these servers are operated by a closed circle of operators. It is very hard for an adversary to add an authority server to the TOR network. "Stable" would at least be of some value for quality of service as those servers are preferred for long living connections, but has not been implemented due to the comparable small increase it would add to the quality of service metric. See subsection 6.2 for further values that might be worth to add in the future. The "named" flag is of no value at all as well as the "hibernating" flag, because TOR tries to minimize the number of "hibernating" nodes by design, so this value is expected to stay low most of the time.

4.1.2 Country numbers

The distribution of servers over different countries and therefore different jurisdictions is the second value that influences the degree of anonymity. With data retention laws, Internet surveillance and other possibilities of law enforcements it is of interest to monitor where the TOR servers are located. If TOR servers are affected by data retention and if this would enforce logging on the TOR servers is still unclear. Especially with the data retention directive from the European Union these numbers might change in the near future, depending on how the European countries implement it and the consequences on the TOR operators. For simplicity only the number of servers and the number of exit servers per country is monitored, the other values mentioned above of fast servers and guard servers would provide only little additional information for monitoring. For detailed analyses they would be interesting though.

To keep the demands on the performance of the monitoring system low and to prevent the user from an incontrollable information flood, only selected countries will be monitored. The countries with the highest number of TOR servers are those of interest. Over a period of two months the country distribution of TOR servers has been monitored, resulting in the implementation of the following set of countries to monitor: United States of America, Germany, China, France, Sweden, Russia, the Netherlands, Canada, Great Britain, Italy and Austria. Additionally the sum of all other countries and TOR nodes that are not mappable to any country get accumulated and stored as well. The mapping of IP addresses to countries was done by using TOR Status, more precisely by using the GeoIP library together with TOR Status.

An excerpt of the collected data is shown in Table 1. It lists the number of TOR servers per country together with the countries top level domains. This list shows some snapshots of the TOR network, the time refers to Central European Time (CET). Similar data has been collected for the number of exit servers per country.

Based on this information, the top 11 countries have been selected for monitoring. This selection can not be considered static as the distribution might change significantly over time. In that case, the country selection and accordingly the implementation has to get adapted. Especially the lower ranks are close together in numbers, which means that depending on the daytime the position varies strongly. Any changes to the implemented ranking should therefore get thoroughly checked over a couple of samplings, and

03.10.08, 16:00	27.10.08, 15:00	21.11.08, 15:00	15.12.08, 11:00
381 .de	334 .us	332 .us	316 .us
333 .us	334 .de	329 .de	295 .de
70 .fr	69 .fr	75 .fr	68 .fr
55 .cn	47 .cn	41 .nl	51 .se
42 .nl	44 .nl	41 .se	46 .cn
41 .se	44 .se	36 .cn	34 .nl
34 .it	32 .it	34 .it	28 .ca
29 .ca	30 .ru	29 .uk	25 .it
29 .uk	25 .uk	27 .ru	24 .ru
27 .at	25 .ca	25 .at	23 .at
26 .ru	24 .at	23 .ca	21 .uk
20 .fi	18 .fi	17 .fi	18 .fi
19 .pl	13 .bg	13 .dk	10 .ch
...	...	...	...

Table 1: Number of servers per country

not only because of one sample.

4.2 Anonymity considerations

Of course, when working on an anonymity network it has to be assured that by adding features the additional features should never possibly harm the network. This means that it should not introduce new attacks or make existing attacks easier.

In our case, all the used information is part of the network consensus, which means that all the informations are already known to every TOR server as well as every TOR client. The only difference is that some coarse part of it is now stored over time. From all the recent attacks on the TOR network (some of them were introduced in subsection 3.1.4), none becomes easier. And even future attacks should not benefit from it, as e.g. the number of servers per country or the total number of exit server are no valuable informations when trying to identify users. The main objective is to provide the interested user an overview over the current status of the underlying TOR network and to make the changes visible.

But as the TOR software is still evolving, care has to be taken when using this information for critical parts of the TOR software (like the path selection algorithm) in the future. How this could happen or why the overall network would benefit from this is discussed in section 6.2.

4.3 Implementation

The gathering of the data was implemented by expanding the TOR Status update script, *tns_update.pl*. Whenever the script is updating the TOR Status database, it is also updating the TOR History monitoring. The values are stored in a RRD, a round robin database, which is also used for generating and updating the other graphs of the TOR Status website. All together, the changes made to *tns_update.pl* were solely for expanding the features of TOR Status, resulting in a convenient and easy integration.

The frontend of the implementation is given by the statistical pictures the PHP file, *network_history.php*, in every category for six different time spans:

- last 6 hours
- last 24 hours
- last week
- last month
- last 3 months
- last year

The time spans are the same for all categories in order to provide a consistent picture to the user. They were chosen to provide recent informations as well as longterm overviews.

5 Data Interpretation

This chapter demonstrates the application of the monitoring and shows the actual collected values and interesting characteristics of the TOR network along with an according interpretation. Note that, as already stated before, this values only reflect the infrastructure of the TOR network and the number of TOR servers, and does not testify anything about the number of users.

The data collection started in the middle of October 2008 and is still going on. The time interval of the collected data presented here refers to around three months. This time windows is large enough for advanced interpretations. All graphics here were created using RRDtool, the same tool which is in use to create the graphics on the TOR Status website.

The represented data samples are customized for this work to illustrate the findings, although closely related to the more elaborate graphics on the TOR Status website. During the data collection period the collection process failed twice for a short period of time, resulting in two small data gaps in some of the graphics. Note that the time zone in the graphics is GMT.

5.1 Overall Network Size

Since deployment, the overall network size and the number of running TOR servers has been steadily increasing. Since public deployment in October 2003 with only a handful of servers, the infrastructure has grown to a total of 50 nodes by November 2004 [33]. This doubled to around one hundred nodes in January 2005 [17], probably (and possibly among other things) because of a presentation about the TOR project and the basic idea of onion routing at the Chaos Communication Congress in Berlin, Germany ("21c3") at the end of December [50]. In August 2007 there were around 1000 running servers as of [53]. Up to today, everyday between 1100 and 1200 servers form the basic infrastructure of the TOR network, whereof 500 to 600 of them are exit servers.

Figure 1 shows the overall number of servers and the number of exit servers during the whole data collection period. On the three month average, 1163 servers were online during the period, 532 of them exit servers. The highest number of servers were 1289, the lowest 1040 (626 /454 exit servers).

Table 2 shows the average numbers extracted from figure 1. The dates represent the times where the process failed and collected no data. As it can

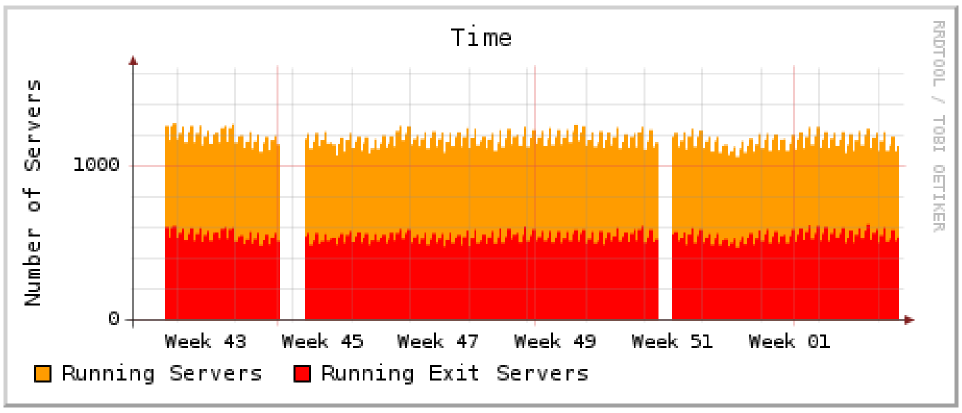

Figure 1: Running TOR Servers in the last 3 months

	Normal	Exit
October (19.-31.)	1185	541
November (4.-30.)	1159	525
December (1.-16)	1174	534
December (18.-31)	1141	523
January (1.-13.)	1159	546

Table 2: Monthly server average

Uptime in weeks	Number of Servers
< 1	549
1	109
2	56
3	38
4	35
5-9	79
10-19	27
> 20	12
Sum:	905

Table 3: Server Uptime Distribution

be seen in figure 1 and table 2, the number of servers stays roughly the same, the network is quite constant.

But this does not mean that these are the same servers all the time: more then half of the servers that report their uptime have an uptime less then one week. Less then five percent have an uptime which is longer then ten weeks. About eighty percent of the servers have a reported uptime in TOR Status. A sample uptime distribution is shown in table 3 from January 13 midday. At that time 1141 servers were marked running in TOR Status.

All in all, the data shows that the TOR network is constantly changing, but that it stays roughly the same size.

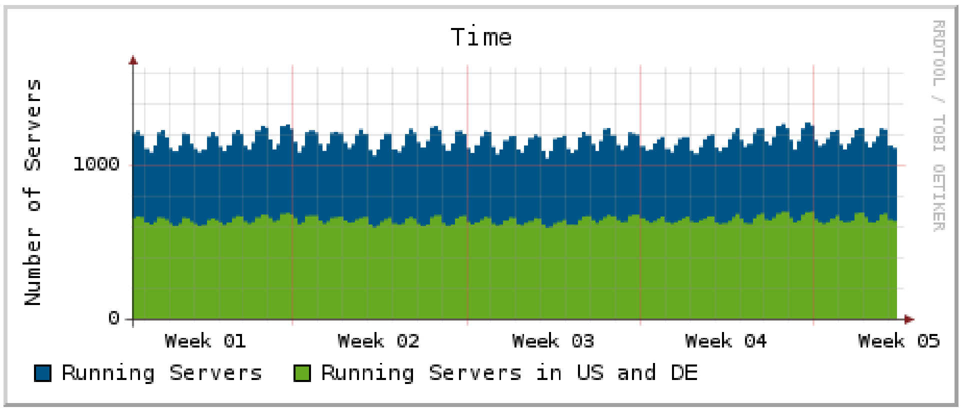

Figure 2: Running Servers in DE and US, in total

When we start looking at the top participating countries, the top countries have already been discussed in subsection 4.1.2. On country level the most active countries are by far Germany and the United States of America. Both countries together host in total more than half of all the servers (this is shown in figure 2). The reasons for that are not easy to identify. Among other reasons, data privacy seems to have a high significance in Germany, probably based on the countries history and because of many popular organizations that want to enforce civil rights and electronic privacy like the Chaos Computer Club. In the US on the other hand, there is a different approach on data privacy compared with e.g. Germany or the European Union, possibly resulting in a higher need for privacy along with the willingness to help others to get it by running a TOR server. Both countries are among the top five countries regarding the Internet density in the world, according to internetworldstats.com, which is also true for the other top TOR countries. If the Internet density is high, Internet access and bandwidth for the users becomes cheap. This of course increases the likelihood that a user is willing to donate bandwidth to the TOR project by running a TOR node.

5.2 Observed Patterns

Due to the distribution of the TOR servers in the world and their specific attributes, special pattern have been found that are quite interesting and characteristic for the TOR network. No other open anonymity service has yet reported comparable patterns, as for example it is much harder to add a server to the JAP network in comparison to the TOR network. Many attributes have an influence on the TOR network, for example where the TOR server is operated (at home, at work or in a data center), if it is an exit server or not and the motivation of the server operator. All together, these pattern can deliver useful informations.

5.2.1 Total number pattern

The most obvious pattern is that the number of overall servers is dependent on the current daytime. This is shown in figure 3 and to some extend in figure 2. It looks like a sine function (or cosine) as the number of servers reaches a maximum and a minimum within 24 hours every day. The maximum and the minimum stays as constant as the average number of servers which is shown in table 4. This table is similar to table 2, except that the minimum and maximum values of the time slots were calculated. It is reasonable to argue that the maximum and the minimum do not say anything about the distribution, but they are important values for the quality of service of the TOR network.

The time in the figure is GMT, which means that most of the servers

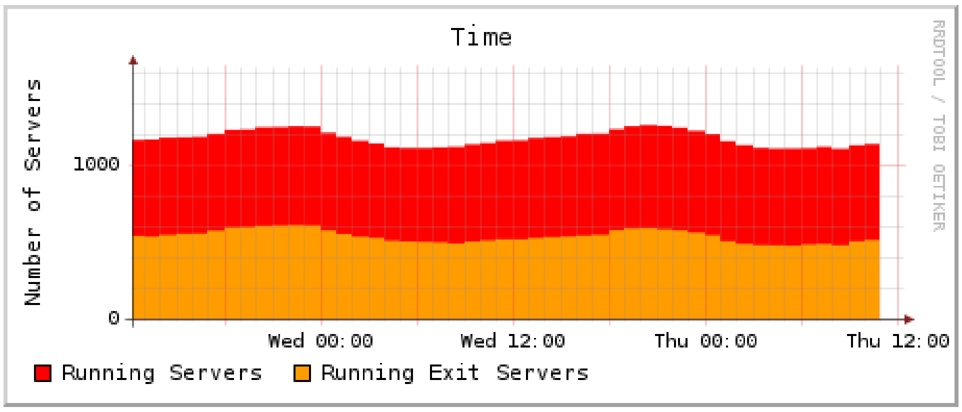

Figure 3: Running TOR servers in the last 2 days

	Normal		Exit	
	min	max	min	max
October (19.-31.)	1076	1289	471	623
November (4.-30.)	1050	1267	459	600
December (1.-16)	1077	1266	471	618
December (18.-31)	1040	1251	454	601
January (1.-13.)	1057	1260	477	626

Table 4: Monthly minimum and maximum number of servers

are online in the "GMT evening" which is insofar astonishing as these data represent the worldwide TOR network. This is mainly influenced (among other things) by the daily variation of servers in Germany, where most of the servers are online in the evening. The number of servers in Germany per day is between 290 and 350. More country specific details and how the countries differ in the daily pattern are discussed in subsection 5.2.2.

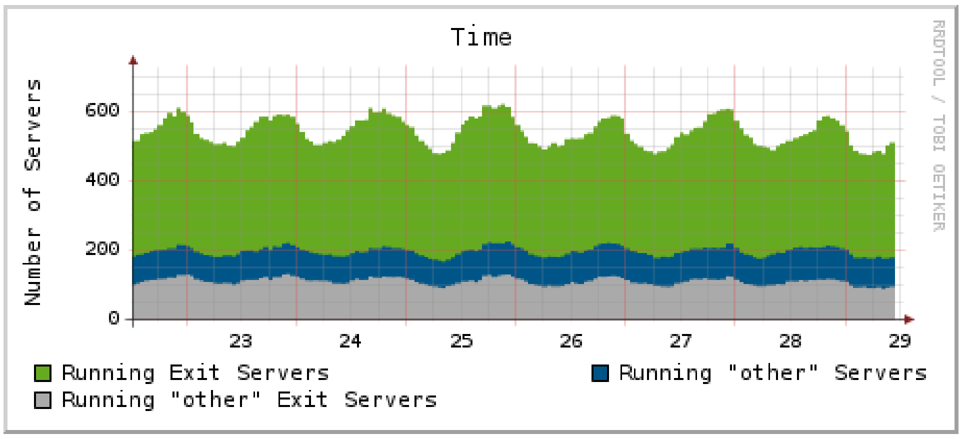

Figure 4: Daily pattern in "other servers"

This daily pattern is also found, strangely enough, in the data set of the "other" servers. These are the servers were the country is unknown, summed up with the servers that are in a different country then the top TOR countries. The top TOR countries can be found in section 4.1.2 and are those where the number of server and exit servers is monitored separately. The sum of the other servers and how it changes over time is shown in figure 4. By looking at it, it is visible that this category of servers as well has a daily pattern. The amplitude is much lower then e.g. compared with the number of exit servers, but it is still visible. This means that, first of all, not only Germany is the reason that the number of servers changes over the day, but also the servers that are not in the top countries. Secondly it seems that these servers have to be geographically close, which is needed that an overall pattern becomes visible.

By looking at the data, two big regions have been identified: the countries in Europe and in Asia. As these countries are on the opposite side of the world, they cancel out their daily pattern as it is shifted by 12 hours. But as the European block has at the moment more servers in total, the pattern is more related to other European patterns and thus has an influence on the overall daily pattern. The most active countries (besides the one that are monitored separately) in Europe are:

- Finland
- Poland
- Switzerland
- Spain
- Greece
- Denmark
- ...

In the Asian region, the following countries host many TOR servers, but not enough to be among the top countries:

- Taiwan
- Hong Kong
- Malaysia
- Japan
- Singapore
- Australia
- ...

These countries together host in total between 15 % and 20 % of the TOR servers. This means, that around 80 % of the servers are closely monitored by the implementation, and that most of the servers come from only a few countries. The proportion of the TOR network located in these countries stays quite constant, which is shown in figure 5.

Finally, regarding the overall network pattern, the proportion of exit servers is of importance. The TOR network is build that 33 % of the servers need to be exit servers. Figure 6 shows that between 45 % and 50 % of the servers are exit servers, which is very good. Just because a server is flagged as exit server does not mean that the client software has to choose this server as the last one. An exit server is also capable of working as the first or second node in the chain of TOR servers. This allows the clients a broad spectrum

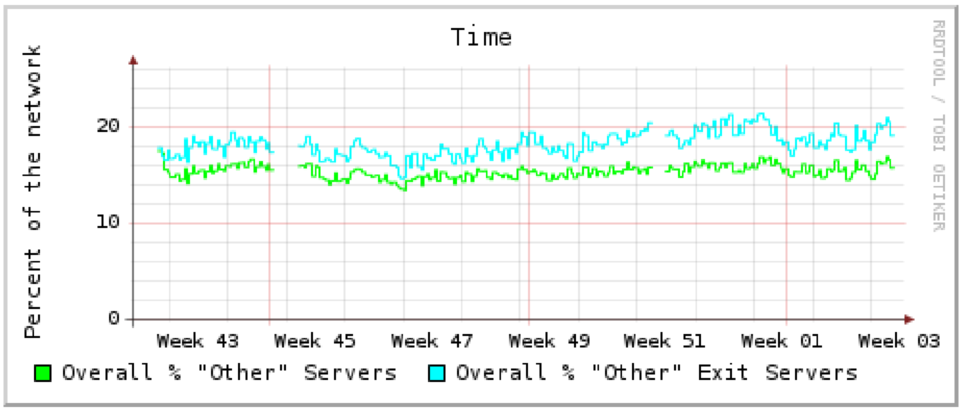

Figure 5: "Other" servers percent of the overall network

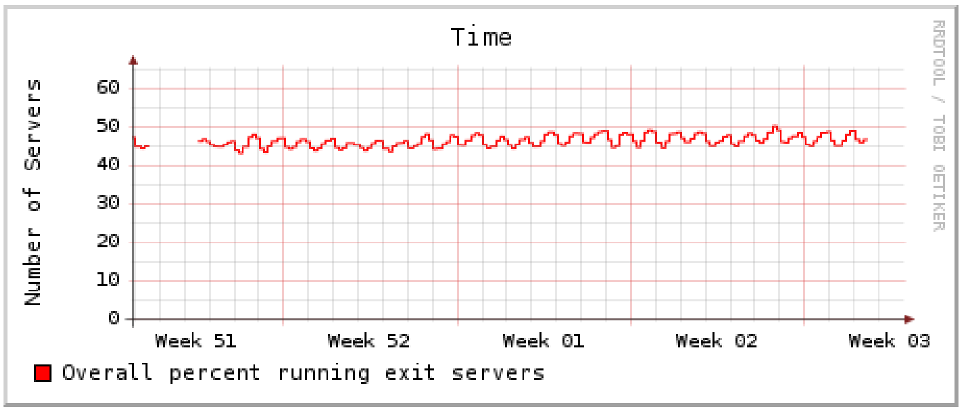

Figure 6: Overall percent exit servers

of possible servers to choose from.

It is also interesting to see that the ratio of normal to exit servers stays quite as constant as the rest of the TOR network. Sudden changes in the percentage of exit servers would be a good indicator in case of any attacks, for example if an attacker tries to redirect as many users as possible to his exit servers by attacking the others with a denial of service attack. This percentage of exit servers does not take into account the various exit server policies, but gives a good overview. In the future it would make sense to take the various exit policies into account as well.

5.2.2 Country pattern

When watching the data on country level, other interesting patterns can be identified. The patterns discussed in this chapter refer to the top countries that are monitored separately, the list can be found in section 4.1.2, respectively in table 1.

The biggest part of the TOR network is made by servers from Germany and the United States of America. But by comparing those two countries, subtle differences are detectable. This is shown in figure 7. While the number

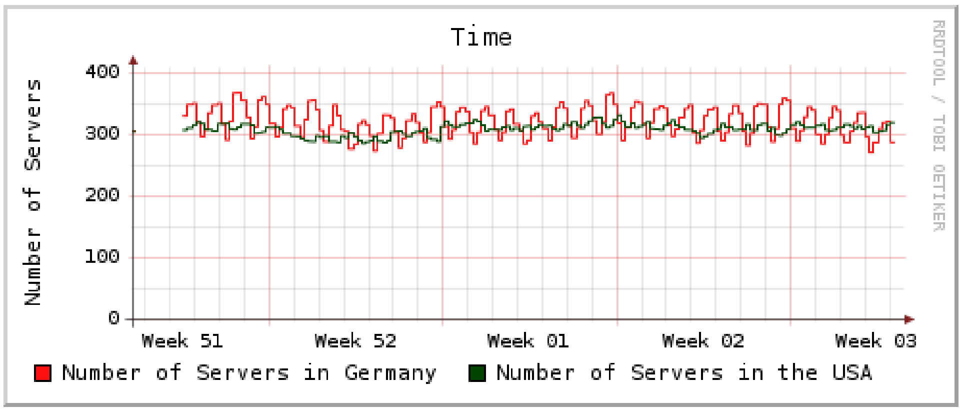

Figure 7: Differences Germany and the USA

of servers in the USA is quite constant, the number of servers in Germany is changing with a pattern within 24 hours. Around 50 servers, or 15 % of the German servers are turned of and on again during the night. This might be because of Internet access pricing models from Internet service providers or because the servers are operated at workplaces, the exact reasons are hard to tell.

The next biggest TOR countries that are comparable in size are China and Sweden. They host each up to 50 servers of the TOR network. Their pattern is shown in figure 8. Notice that compared to figure 7 the numbers are much smaller, resulting in bigger amplitude changes. Both countries

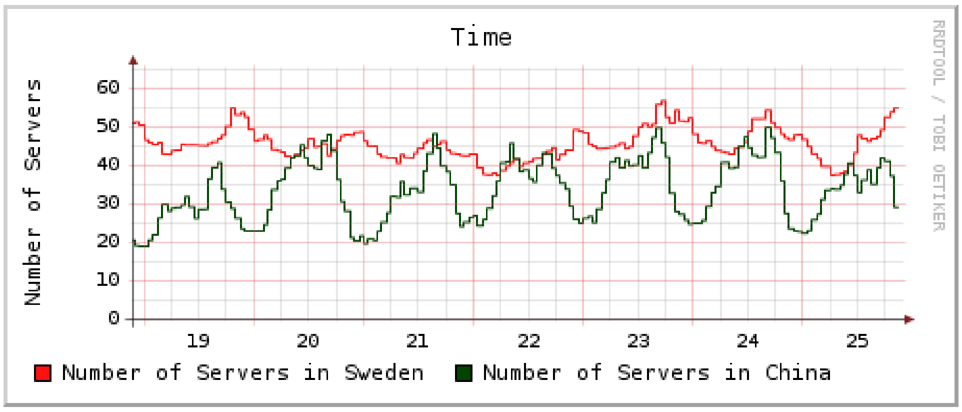

Figure 8: Differences Sweden and China

have a daily pattern, while the change in numbers in China is stronger. This might be because bandwidth might be more expensive in China, as Sweden is one of the European countries with the lowest costs for bandwidth. As both countries are quite on the opposite side of the world, the daily pattern is more closely related as the expected naive mirror inverted relation. This means that the peak in the number of servers is reached in different daytimes.

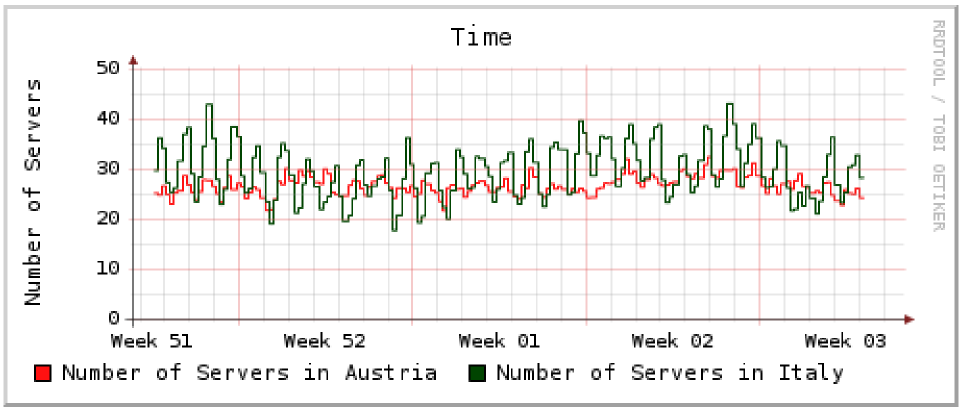

Figure 9: Differences Austria and Italy

Comparable in size are also Austria and Italy. They are neighboring countries and have roughly the same number of TOR servers. Their graph is shown in figure 9. The fluctuation of daily servers is stronger in Italy, while in that sample there is no daily pattern for Austria. But as Austria only has around 30 TOR servers, this is probably not enough to show a clear daily pattern.

Last but not least we will show how many servers per country are exit servers. This is shown in figure 10. Only the top 5 countries have been

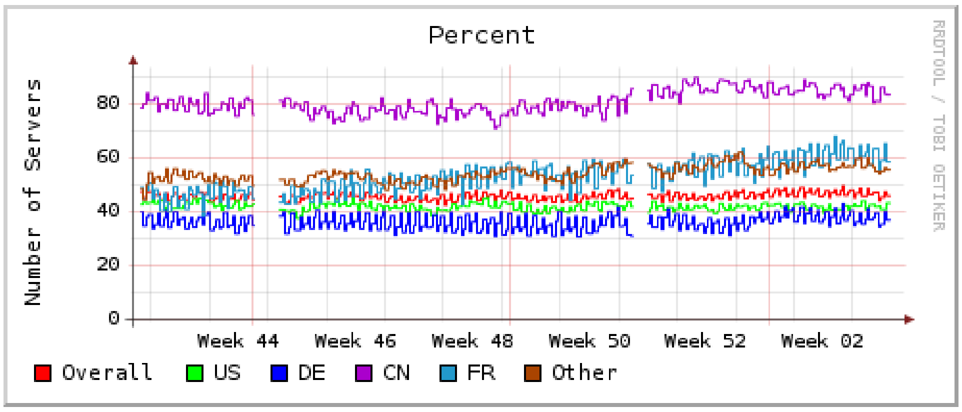

Figure 10: Percent of Exit Servers

compared, for simplicity of the graph. It is very interesting to see which countries have more than average exit servers, the average was already shown in figure 6. Germany and the United States of America have less then average percentage of exit servers, while the smaller countries China and France have a higher percentage, most likely because of the low numbers. Chinas has between 80 and 90 % exit servers.

5.3 Strange Pattern found

During the collection of data, some strange pattern have been found. One of them seems to be inside of TOR itself: When looking at the number of guard servers, the values change quite a lot. It seems as if the TOR consensus is adding and removing a large proportion of guard servers. This is shown in figure 11. Although the number of guard servers seems to change a lot, there is still a daily pattern visible. The sine pattern again can be attributed to Germany, the country with the highest number of guard servers and some other countries. But the large jumps, most of the time more then 100 guard nodes, is not explicable by daily pattern. It is likely to be due to some topics in the TOR implementation, or the network consensus protocol. The analysis of this pattern is beyond the scope of this thesis, the issue has been reported

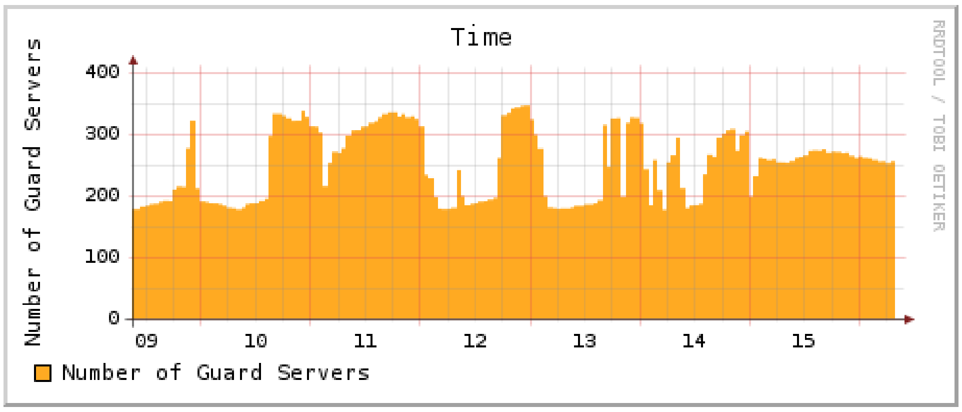

Figure 11: Problem with number of Guard Servers

to the TOR developer team.

A possible security related incident is shown in figure 12. Around Christmas

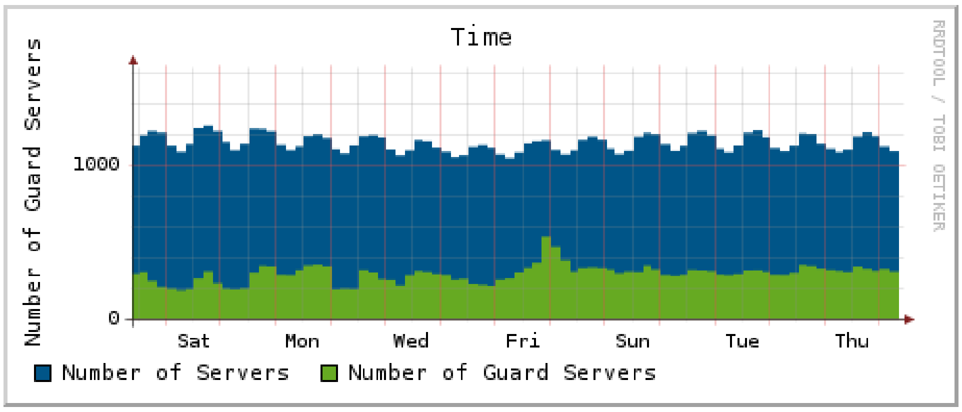

Figure 12: Peak of Guard Servers

2008, more exactly on the 26th of December, a sudden peak in the number of guard servers occurred. All of the sudden more then 500 servers were labeled as guard nodes. This is a lot compared with the normal number which is normally between 200 and 300. This could have been because of a specific behaviour or bug in the implementation of the guard flag, or it could also have been an attack. It is possible that an attacker was trying to set up as many guard nodes as possible so that many new users choose one of the manipulated guard servers as fixed entry node. However, the number of running servers did not change and there was no discussion on the TOR mailing list, which indicates that this incident was not an attack but more likely caused by the behaviour of the network consensus.

6 Conclusion

With the described monitoring of the TOR network several interesting patterns and characteristics of the TOR network have been found. These patterns can be used in the future to improve the overall TOR network and to detect and analyse attacks on it. The implementation itself was incorporated into TOR Status and will be published with the next software release to allow widespread use. Alternatively they can be already found in the developer subversion repository of TOR Status.

6.1 Overall benefits

The overall benefits of the monitoring to the TOR community are manifold. First, the most obvious benefit is that the development and growth of the TOR network is now visible to the TOR developers, the server operators, the users and everyone interested. The information is presented conveniently and self-explanatory, no deep knowledge or understanding of the TOR network is needed. This feedback can be used by the advanced user to decide about the current degree of anonymity and whether the network is providing sufficient anonymity or not. One example would be a user who wants to use the TOR network only at times where there is a peak in the overall number of servers, for maximum anonymity. Another user might seek as much performance as possible, thus trying to find a country with as many servers as possible and using TOR when it is in the middle of the night in that country.

These kinds of manual “path improvements” can also eventually get incorporated into the TOR path selection algorithm. An ongoing discussion is whether the user should or might choose between the performance and the provided anonymity (see subsection 3.1.5). The collected statistics about the history of the TOR network would allow decisions which country to use as entry to TOR. Consider for example that all of the sudden the number of servers in for example Italy drops significantly. Is it then still ok to use the servers in Italy? It probably is ok for a user relatively close to Italy who is seeking high data throughput, if the Italian servers are used as middleman or entry node. For a user seeking high anonymity it might be better to avoid Italy in all circuits at all. By comparing the present with the past it becomes possible to observe trends and this might be a triggering point for an analysis or discussion.

This brings us to a very important benefit, the possibility of attack detection. Right now, only the operator of a server is able to detect attacks on servers under his control, like denial of service attacks. Sometimes this is discussed on the TOR mailing list, `or-talk@freehaven.net`, mainly to check if other operators have similar problems. If an adversary would launch a more significant attack, for example by a distributed denial of service attack on half of the TOR servers, the real impact on the TOR infrastructure would be hard to detect fast. The same holds if an adversary launches a Sybil attack, by adding numerous probably manipulated servers. With the monitoring of the TOR network this becomes easier to detect, as the change in the number of servers is reflected immediately.

6.2 Further improvements

The implementation provides a basic monitoring of the TOR network and leaves room for improvements and feature enhancements in the future. The collected data set might get expanded to collect information of more countries, and collect additional data on the overall network. One example would be the stable flag, which is used for long living connections on certain ports. As this flag does not influences anonymity and performance the information gain would be considerably low. Other informations worth collecting are the total transmitted traffic (as in [17]), but as this information is provided solely by the servers it is easy to manipulate and to bias the statistics.

Another feature worth implementing would be the port distribution of the exit servers and the country distribution of specific important ports like HTTP. This would allow the definition of "optimistic" and "pessimistic" server operators by looking at the servers exit policy. As the focus of this work was on basic network monitoring this is for future investigations.

- Optimistic server operators are allowing most of the ports, denying only certain, probably known bad ports like file sharing ports. This is often referred to as "blacklisting".
- Pessimistic server operators allow only certain ports of personal or overall importance, like SSH, HTTP or HTTPS. This is referred to as "whitelisting".

In the far future, the collected information could be used to compare the current state of the network with the expected state. By calculating a conservative set of rules about the "normal" behaviour of the TOR network, changes that could probably have an influence on the security can get detected. Incorporating these rules into the path selection algorithm would allow fine grained privacy constraints on the selection of nodes. An example would be an adaptive path length: in the normal case, three nodes could get selected as it is currently implemented. If then all of the sudden the number of servers drops, or significantly increases, the path length could be automatically set to e.g. six TOR nodes to increase security. It is though not clear if an increase in the path length would increase the security, but it is sufficient as an example where the collected data of the TOR network could get used in the future. Another example would be the integration of the data into TORFlow, a project to detect malicious nodes. Instead of testing all the servers all the time, which could be eventually costly in terms of time and resources, only selected TOR nodes or all the nodes in a specific country could get tested. Based on the collected data a set of TOR nodes responsible for the most suspicious changes in the infrastructure could get identified for testing.

However, extensions should be done only very carefully and only after evaluating the impact of the measure on anonymity and possible attacks.

List of Tables

List of Figures

References

[1] Adam Back, Ulf Möller, and Anton Stiglic. Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems. In *Proceedings of Information Hiding Workshop (IH 2001)*, pages 245–257, April 2001.

[2] Salman Baset and Henning Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. *CoRR*, 2004.

[3] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-Resource Routing Attacks Against Tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007)*, October 2007.

[4] Oliver Berthold and Hannes Federrath. Project "anonymity and unobservability in the internet". *Workshop on Freedom and Privacy by Design / CFP2000*, 2000.

[5] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In *Preproceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129, 2000.

[6] Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom Systems 2.0 Architecture. White paper, Zero Knowledge Systems, Inc., December 2000.

[7] R. T. Braden. Requirements for Internet hosts — communication layers. RFC 1122 (Standard), 1989. `http://www.ietf.org/rfc/rfc1122.txt`.

[8] Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP Covert Timing Channels: Design and Detection. In *CCS '04: Proceedings of the 11th ACM conference on Computer and Communications Security*, pages 178–187, 2004.

[9] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[10] George Danezis. Introducing Traffic Analysis: Attacks, Defences and Public Policy Issues. Technical report, University of Cambridge Computer Lab, 2005.

[11] George Danezis and Claudia Diaz. A Survey of Anonymous Communication Channels. Technical report, Microsoft Research, January 2008.

[12] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 2–15, May 2003.

[13] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[14] Roger Dingledine and Nick Mathewson. Design of a blocking-resistant anonymity system (DRAFT). `https://svn.torproject.org/svn/tor/trunk/doc/design-paper/blocking.pdf`.

[15] Roger Dingledine and Nick Mathewson. Anonymity Loves Company: Usability and the Network Effect. In *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*, June 2006.

[16] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[17] Roger Dingledine, Nick Mathewson, and Paul Syverson. Challenges in deploying low-latency anonymity (DRAFT), February 2005. `https://www.torproject.org/svn/trunk/doc/design-paper/challenges.pdf`.

[18] John R. Douceur. The Sybil Attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 2002.

[19] David Goldschlag, Michael Reed, and Paul Syverson. Onion Routing for Anonymous and Private Internet Connections. *Communications of the ACM*, 42:39–41, 1999.

[20] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In *Information Hiding*, pages 137–150, 1996.

[21] Andrew Hintz. Fingerprinting Websites Using Traffic Analysis. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*, April 2002.

[22] Stefan Köpsell. AnonDienst - Design und Implementierung, 2004.

[23] Olaf Landsiedel, Alexis Pimenidis, Klaus Wehrle, Heiko Niedermayer, and Georg Carle. Dynamic Multipath Onion Routing in Anonymous

Peer-To-Peer Overlay Networks. In *Proceedings of the GLOBECOM 2007*, 2007.

[24] David Martin and Andrew Schulman. Deanonymizing users of the SafeWeb anonymizing service. In *Proceedings of the 11th USENIX Security Symposium*, 2002.

[25] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas C. Sicker. Shining Light in Dark Places: Understanding the Tor Network. In *Privacy Enhancing Technologies*, pages 63–76, 2008.

[26] Damon McCoy, Kevin Bauer, Dirk Grunwald, Parisa Tabriz, and Douglas C. Sicker. Shining Light in Dark Places: A Study of Anonymous Network Usage. Technical report, University of Colorado at Boulder, 2007.

[27] Adam Meyerson and Ryan Williams. General k-Anonymization is Hard. Technical report, In Proc. of PODS'04, 2003.

[28] Mixminion network load. `http://www.noreply.org/load/`.

[29] Number of Mixminion Nodes. `http://www.noreply.org/mixminion-nodes/`.

[30] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Standard), 1987. `http://www.ietf.org/rfc/rfc1034.txt`.

[31] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. Draft, July 2003. `http://www.abditum.com/mixmaster-spec.txt`.

[32] Steven J. Murdoch. Hot or Not: Revealing Hidden Services by their Clock Skew. In *Proceedings of CCS 2006*, October 2006.

[33] Steven J. Murdoch and George Danezis. Low-Cost Traffic Analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.

[34] Steven J. Murdoch and Robert N. M. Watson. Metrics for Security and Performance in Low-Latency Anonymity Networks. In *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, July 2008.

[35] Lasse Overlier and Paul Syverson. Locating Hidden Servers. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 100–114. IEEE Computer Society, 2006.

[36] Lasse Øverlier and Paul Syverson. Improving Efficiency and Simplicity of Tor Circuit Establishment and Hidden Services. In *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, June 2007.

[37] Andriy Panchenko, Lexi Pimenidis, and Johannes Renner. Performance Analysis of Anonymous Communication Channels Provided by Tor. In *ARES '08: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*, pages 221–228. IEEE Computer Society, 2008.

[38] Sameer Parekh. Prospects for Remailers. *First Monday*, 1(2), August 1996. `http://www.firstmonday.dk/issues/issue2/remailers/`.

[39] Andreas Pfitzmann and Marit Hansen. Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology. Draft, May 2003. `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml`.

[40] Andreas Pfitzmann and Marit Hansen. Anonymity, Unobservability, and Pseudonymity: A Consolidated Proposal for Terminology. Draft, February 2008. `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml`.

[41] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, February 1991.

[42] Birgit Pfitzmann and Andreas Pfitzmann. How to break the direct RSA-implementation of MIXes. In *Advances in Cryptology—EUROCRYPT '89 Proceedings*, pages 373–381, 1989.

[43] J. Postel. Internet Protocol. RFC 791 (Standard), 1981. `http://www.ietf.org/rfc/rfc791.txt`.

[44] Steven J. Murdoch Richard Clayton and Robert N. M. Watson. Ignoring the Great Firewall of China. In *Privacy Enhancing Technologies*, pages 20–35, 2006.

[45] Stephen Rollyson. Improving Tor Onion Routing Client Latency. Technical report, Georgia Tech College of Computing, 2006.

[46] Robin Snader and Nikita Borisov. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Proceedings of the Network and Distributed Security Symposium - NDSS '08*, February 2008.

[47] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on SSH. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, 2001.

[48] Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. *International journal of uncertainty, fuzziness, and knowledge-based systems*, 2002.

[49] Debian Security Team. Debian OpenSSL Security Advisory: DSA-1571-1 openssl, May 2008. `http://www.debian.org/security/2008/dsa-1571`.

[50] TOR presentation at the 21st Chaos Communication Congress in Berlin, 2004. `http://www.ccc.de/congress/2004/fahrplan/event/183.en.html`.

[51] TOR Path Specification. `https://svn.torproject.org/svn/tor/trunk/doc/spec/path-spec.txt`.

[52] TOR Network Status Project. `http://project.torstatus.kgprog.com/`.

[53] TOR Running Routers, outdated as of August 2007. `http://www.noreply.org/tor-running-routers`.

[54] Nart Villeneuve. Breaching Trust: An analysis of surveillance and security practices on China's TOM-Skype platform, 2008.

[55] Wei Wang, Mehul Motani, and Vikram Srinivasan. Dependent link padding algorithms for low latency anonymity systems. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 323–332, 2008.

[56] Jonathan Zittrain and Benjamin Edelman. Internet Filtering in China. *IEEE Internet Computing*, 7(2):70–77, 2003.