Traffic Analysis Attacks and Defenses in Low Latency Anonymous Communication

Sambuddho Chakravarty

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2014

ABSTRACT

Traffic Analysis Attacks and Defenses in Low Latency Anonymous Communication

Sambuddho Chakravarty

The recent public disclosure of mass surveillance of electronic communication, involving powerful government authorities, has drawn the public's attention to issues regarding Internet privacy. For almost a decade now, there have been several research efforts towards designing and deploying open source, trustworthy and reliable systems that ensure users' anonymity and privacy. These systems operate by hiding the true network identity of communicating parties against eavesdropping adversaries. Tor, acronym for *The Onion Router*, is an example of such a system. Such systems relay the traffic of their users through an overlay of nodes that are called *Onion Routers* and are operated by volunteers distributed across the globe. Such systems have served well as anti-censorship and anti-surveillance tools. However, recent publications have disclosed that powerful government organizations are seeking means to de-anonymize such systems and have deployed distributed monitoring infrastructure to aid their efforts.

Attacks against anonymous communication systems, like Tor, often involve *traffic analysis*. In such attacks, an adversary, capable of observing network traffic statistics in several different networks, correlates the traffic patterns in these networks, and associates otherwise seemingly unrelated network connections. The process can lead an adversary to the source of an anonymous connection. However, due to their design, consisting of globally distributed relays, the users of anonymity networks like Tor, can route their traffic virtually via any network; hiding their tracks and true identities from their communication peers and eavesdropping adversaries. De-anonymization of a random anonymous connection is hard, as the adversary is required to correlate traffic patterns in one network link to those in virtually all other networks. Past research mostly involved reducing the complexity of this

process by first reducing the set of relays or network routers to monitor, and then identifying the actual source of anonymous traffic among network connections that are routed via this reduced set of relays or network routers to monitor. A study of various research efforts in this field reveals that there have been many more efforts to reduce the set of relays or routers to be searched than to explore methods for actually identifying an anonymous user amidst the network connections using these routers and relays. Few have tried to comprehensively study a complete attack, that involves reducing the set of relays and routers to monitor and identifying the source of an anonymous connection. Although it is believed that systems like Tor are trivially vulnerable to traffic analysis, there are various technical challenges and issues that can become obstacles to accurately identifying the source of anonymous connection. *It is hard to adjudge the vulnerability of anonymous communication systems without adequately exploring the issues involved in identifying the source of anonymous traffic.*

We take steps to fill this gap by exploring two novel active traffic analysis attacks, that solely rely on measurements of network statistics. In these attacks, the adversary tries to identify the source of an anonymous connection arriving to a server from an exit node. This generally involves correlating traffic entering and leaving the Tor network, linking otherwise unrelated connections. To increase the accuracy of identifying the victim connection among several connections, the adversary injects a traffic perturbation pattern into a connection arriving to the server from a Tor node, that the adversary wants to de-anonymize. One way to achieve this is by colluding with the server and injecting a traffic perturbation pattern using common traffic shaping tools. Our first attack involves a novel remote bandwidth estimation technique to confirm the identity of Tor relays and network routers along the path connecting a Tor client and a server by observing network bandwidth fluctuations deliberately injected by the server. The second attack involves correlating network statistics, for connections entering and leaving the Tor network, available from existing network infrastructure, such as Cisco's NetFlow, for identifying the source of an anonymous connection. Additionally, we explored a novel technique to defend against the latter attack. Most research towards defending against traffic analysis attacks, involving transmission of dummy traffic, have not been implemented due to fears of potential performance degradation. Our novel technique involves transmission of dummy traffic, consisting of packets

with IP headers having small Time-to-Live (TTL) values. Such packets are discarded by the routers before they reach their destination. They distort NetFlow statistics, without degrading the client's performance. Finally, we present a strategy that employs transmission of unique plain-text decoy traffic, that appears sensitive, such as fake user credentials, through Tor nodes to decoy servers under our control. Periodic tallying of client and server logs to determine unsolicited connection attempts at the server is used to identify the eavesdropping nodes. Such malicious Tor node operators, eavesdropping on users' traffic, could be potential traffic analysis attackers.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I take this opportunity to thank my research advisor, Professor Angelos D. Keromytis, for his constant support, suggestions and encouragements during this odyssey. Even though this journey was filled with learning and growth, I would say I am just starting to learn. The struggle, trials and tribulations of life as a graduate student are not merely those that involve technical and intellectual obstacles. The mind presents far more challenging psychological and emotional hurdles to oneself, in more ways than one ever can conceive. Professor Keromytis has been extremely patient and understanding throughout my ordeals in trying to tackle such hurdles.

I am totally indebted to my father, mother, all other family members and all my teachers since kindergarten, for their love, support and encouragement throughput my entire educational process. They have continuously encouraged me and nurtured my curiosity to learn more. They have have been perennial fountains of inspiration.

I am very grateful to all my co-worker in all the projects that I undertook – Angelos Stavrou, Michalis Polychronakis, Georgios Portokalidis, Marco V. Barbera and Gundeep S. Bindra, for all their ideas, support, help and encouragement.

I am very much thankful to all my colleagues and lab mates for their ideas, support, help and encouragement.

I am also very thankful to all CRF members for their help and support that involved providing me the necessary hardware and software resources, as and when needed.

I am also extremely thankful to all my friends for their love, support and encouragement all these years.

Finally I thank all the committee members for devoting their time to read this thesis and providing me with their useful feedback.

Chapter 1

Introduction

1.0.1 Introduction

The changes in the computing world in the last few years have dramatically impacted our lives and societies. The popularity of social media has blurred the gap separating private and unrestricted information about individuals. Private information about anyone is now just a few clicks away. *Exactly what information is exposed about an individual and how it could be abused by others?, is not merely a question of debate in various circles but also a vital question concerning every citizen.* On the other hand, socio-economic issues, leading to political uprisings at various places in the Middle East, Egypt and New York, have leveraged such technological advances for various purposes, such as disseminating information, and organizing protest demonstrations. The US Department of State has labelled the Internet as the "Che Guevara of the 21st century" [Ross, 2011]. Repressive government agencies thus try to restrict users' access to such Internet based services [Park and Crandall, 2010; Price and Enayat, 2012]. To counter such censorship measures, and to ensure privacy and anonymity in general, people have resorted to using anonymous communication systems such as Tor [Dingledine *et al.*, 2004], JAP [JAP,], Anonymizer [Anonymizer,], I2P [i2p,] and Mixminion [Danezis *et al.*,]. These systems try to hide the actual identity (IP address) of one (or both) of the communicating parties from their peers, and from adversaries who can eavesdrop on the traffic flowing through the network.

There are two broad categories of anonymous communication systems – *low-latency*

anonymous communication systems, that are geared towards latency sensitive applications, like web browsing, and *high-latency* anonymous communications systems, designed for delay tolerant applications like e-mail and peer-to-peer file transfer. Presently, Tor, the most popular, volunteer operated, low-latency anonymity system serving about half a million users [Tor Metrics Portal,], is considered a de-facto standard for low-latency anonymous communication systems. Famous whistle-blowing systems like `wikileaks` [Assange,], rely on Tor for enabling anonymized submission of sensitive documents. Systems like JAP and I2P also fall under this category. Others, such as Mixminion, an anonymous e-mail system, and OneSwarm [Isdals *et al.*, 2010], an anonymous file sharing system, fall under the category of high-latency anonymous communication systems.

The origins of most anonymization systems can be traced to the architecture presented by David Chaum in his seminal paper describing a scheme to transmit untraceable e-mails [Chaum, 1981]. In his scheme, data is transmitted between communicating peers via a cascade of proxies (known as anonymization proxies or *Onion Routers*). Before the communication progresses, the communication initiator establishes shared secret keys with the proxies. The communication initiator encrypts the data in layers, starting with the key it shares with the last node on path of proxies, then in a reverse sequential order using the keys it shares with the intermediate nodes (starting with the penultimate node) and finally using the key it shares with the first node of the path. These encrypted messages are then transmitted to the first node of the path, which decrypts them and determines the next hop in the path and sends the messages to it. The second node repeats this process and sends the messages to the third node. This process is repeated for all the nodes along the path. Finally the last node in the path decrypts the first layer of encryption and sends the decrypted original messages to their intended destinations. *Thus, no node along the path can determine both the original sender and the actual intended receiver of the messages*. This process ensures anonymity against the peer as well as eavesdropping adversaries that might snoop on the traffic.

Systems like Tor, have served well to circumvent censorship by powerful adversaries, that can eavesdrop on the network traffic. However, in the recent past, it has been shown that even such systems are vulnerable to detection (and subsequent censorship) [Winter and

Lindskog, 2012] using Deep Packet Inspection (DPI). DPI involves an adversary inspecting real-time traffic for specific byte sequences, corresponding to connections for anonymization proxies, and subsequently filtering such connections. Censorship is, however, one of the problems that users encounter while communicating through anonymous communication systems. The other threat is that of identification of the anonymous users, through an attack on the anonymous communication system itself. Low-latency anonymous communication systems, optimized for performance, strive to transport data between peers by minimizing the transmission delays. The routing process of anonymization relays tries to ensure least degradation of users' traffic performance characteristics, such as throughput and round trip time (RTT). Thus, variations in traffic characteristics at one end of an anonymization path, get replicated to the other end. This makes them vulnerable to *timing attacks* (also known as *traffic analysis attacks* [Raymond, 2001; Raymond, 2000; Wright *et al.*, 2002]). In such attacks an adversary, has a fairly wide view of the anonymization network. He or she can observe traffic entering and leaving several network links (often those that transport many connections). Such an adversary correlates network statistics in these different links and associate otherwise seemingly unrelated network connections and in this way can identify the source of an anonymous connection.

The de-anonymization process, using traffic analysis, generally involves identification of a specific anonymous user[1], among several competing network connections. A traffic analysis adversary having access to the traffic statistics of the network connections entering and leaving the anonymization relays, could correlate seemingly unrelated network connections and links anonymously communicating parties. However, it is believed that it is difficult for even the most powerful adversary to observe the traffic flowing to every possible host in the Internet (a possible Tor user) in order to launch a traffic analysis attack for de-anonymizing an anonymous client; it is akin to the problem of "finding a needle in a haystack". In their original paper [Dingledine *et al.*, 2004], describing the design of Tor, the authors stated that there are very few adversaries who are powerful enough to observe traffic in all the network links and can correctly correlate the traffic characteristics in appropriate links to associate communicating peers. However, several research efforts have been pro-

[1]Possibly communicating to a host that serves sensitive information.

posed to resolve this problem. The de-anonymization process, as it appears from research conducted in the past several years, can be conceived as a two-step process. The first step involves finding the anonymization relays or network links, that very likely transport the traffic to and from the source of an anonymous connection. This step effectively reduces the number of connections that an adversary needs to inspect. The second step generally involves launching a traffic analysis attack to identify the source of an anonymous traffic within this reduced set of relays or network links to be monitored.

Researchers have adopted several strategies for finding the small set of relays or routers carrying the traffic victim traffic (*i.e.*, the first step of the de-anonymization process). Until recently, a common tactic exploited by several researchers, involved launching a form of *Sybil* attack [Douceur, 2002] that involved attracting large number of Tor users to use few malicious Tor nodes that collude together to launch a traffic analysis attack. Tor relays advertising higher bandwidth are likely to attract more clients than others due to a Tor client's relay selection process [Tor Path Specifications,][2]. To optimize users' performance, Tor client's relay selection algorithm selects relays, biased on their advertised bandwidths. Relays advertising higher bandwidth are more likely to be selected in Tor circuits than the ones advertising lower bandwidth. A relay's bandwidth advertisement is configured by its operator. Until recently, these bandwidth advertisements were blindly trusted by the Directory Services. Malicious relay operators could launch a small number of relays and deliberately advertise large bandwidths to attract large number of Tor users; in this way, the relays operators could launch traffic analysis attacks against these users. This observation was first presented by Bauer et al. [Bauer *et al.*, 2007] in their seminal paper and later explored in other attack strategies [Øverlier and Syverson, 2006; Fu and Ling, 2009].

Several researchers have also explored the feasibility of launching de-anonymization attacks by monitoring traffic at Internet Autonomous Systems (ASes) [Feamster and Dingledine, 2004; Edman and Syverson, 2009a]. In their efforts they relied on the observation that ASes common to network paths leading to and from the networks hosting the Tor relays, that advertised higher bandwidth, are more likely to de-anonymize a larger fraction of

[2]Before building a path, Tor relays consult a set of special relays, called *Directory Services* to determine the relays running in the network and obtain information about their advertised bandwidths.

Tor traffic compared to others.

Besides these, there have also been some efforts to use traffic analysis attacks to identify the Tor relays involved in anonymous connections. The first such attack was proposed by Murdoch and Danezis [Murdoch and Danezis, 2005]. Launched on a fledgling Tor network which consisted of less than 50 relays, the attack involved a malicious Tor client that used RTT measurement probes and collusion with the server to determine relays involved in Tor paths. However, it was later demonstrated by Evans et al. [Evans *et al.*, 2009], that such an attack was no longer directly feasible due to the large number of Tor nodes and network congestion.

The second step of the de-anonymization process involves identifying the actual source of an anonymous connection, among the connections that use the anonymization nodes or routers, obtained from the first step of the de-anonymization process. *A careful study of the literature in this area reveals that there have been very few efforts to study the practical effectiveness of launching traffic analysis using network statistics for determining the source of anonymous connections.* Hopper et al., in their paper titled "How Much Anonymity does Network Latency Leak?" [Hopper *et al.*, 2007; Hopper *et al.*, 2010], presented a practical technique to de-anonymize anonymous communications. The authors looked into an effective method to find the actual source of anonymous traffic, using network statistics. They combined the Murdoch Danezis attack with the Vivaldi [Dabek *et al.*, 2004] network coordinate system to estimate the source of anonymous traffic. With moderate accuracy they were able to estimate the location of the client. Their approach however assumes that two network connections might be originating from the same source network if the end-to-end delays of the Tor paths, connecting the clients to the server, are of comparable orders of magnitude. However, these assumptions might not hold valid in today's Tor network which has over 4500 relays and serves over half a million users.

Among the other efforts to de-anonymize anonymous traffic, few have explored the second phase [Murdoch and Zieliński, 2007]. In their paper, the authors describe how a small number of Internet Exchanges (IXes) (physical locations where Autonomous Systems (ASes) connect with one another), can observe traffic entering and leaving a large number of Tor relays and can thus use NetFlow [Claise,], a network monitoring framework installed in

Cisco routers to aid the de-anonymization of anonymous Tor traffic. Their simulation results however shed little light on the question of how useful such systems are for practically de-anonymizing anonymous communication.

During summer and fall of 2013, information published about the activities of organizations such as the National Security Agency (NSA) [National Security Agency/Central Security Service, ; Schneier, 2013] have verified that powerful adversaries, previously assumed hypothetical from a research point of view, are now in fact, close to being able to monitor and de-anonymize anonymous users. They have deployed systems consisting of host monitoring the networks in various places across the globe, in key network locations in order to launch man-in-the-middle attacks by responding to DNS and web requests well in advance of the expected responders.

Past research has focused primarily on the first part of the problem, namely finding the appropriate network links to monitor. The second part of the problem, involving identification of the source of anonymous traffic amidst the several flows, has not been explored enough. Without adequate information about the accuracy of finding the anonymous source and the practical challenges involved in such attacks, opinions regarding the vulnerability of low-latency anonymity network against traffic analysis, remain incomplete.

This is where we posit our research. In our research we focussed on the accuracy, feasibility, and practical problems involved in identifying the source of anonymous traffic amidst several competing network connections, relying solely on measurements of network statistics. We explored such challenges through two novel traffic analysis attacks to identify the source of anonymous traffic by solely observing network statistics and patterns. These statistics are prone to losses and inaccuracies due to network congestion. We also explored practical ways to defend against one of the attacks. Unlike previously proposed solutions for defending against traffic analysis attacks, our strategy to defend against attacks does not degrade users' performance. Moreover, there have been little or no efforts to identify the potential traffic analysis adversaries. We devised a system which can be used to identify such adversaries. Our efforts are summarized in the next subsection.

1.0.2 Summary of Attacks

Our first attack relies on a novel remote bandwidth estimation tool, called *LinkWidth* [Chakravarty *et al.*, 2008c], to verify the identity of the anonymous relays and routers along a Tor *circuit*[3]. In our research, we assumed a powerful adversary which controls an array of nodes with high speed network access, running copies of LinkWidth. This adversary tries to verify the identity of relays and routers that are possibly involved in an anonymous communication session, by correlating the simultaneous changes in their available network bandwidth whenever the communication progresses. The process of remotely estimating available bandwidth, is prone to inaccuracies, primarily due to inadequate bandwidth and background congestion. Thus, to improve our chances of correctly identifying the relays and routers involved in an anonymous connection, we assume that the attacker deliberately injects a traffic perturbation pattern in the anonymous connection and simultaneously observes these perturbations manifest (using LinkWidth) on the relays and routers, leading to the source of the anonymous connection. One way for the adversary to inject such traffic perturbations is to control the server. Thus, the server injects such bandwidth perturbations, on behalf of the adversary using common traffic shaping tools. One may assume that the server hosts content that may be of interest to potential Tor users; such as information that is of interest to dissidents. A Tor client could find such servers through popular search engines. The server may host data such as decoy multimedia files (*e.g.*, pirated films) or may simply inject fake traffic along with regular HTTP responses, simply to congest the network links for a relatively long duration of time (about 5 minutes) and aid the attacker in carrying out the measurements.

It is not essential for the adversary to controls the server. The adversary could employ covert mechanisms to inject such traffic perturbation patterns, without the server knowing about them. In fact, as described in Chapter 3, with little variations, such an approach could be used to verify the identity of the routers connecting Tor *Hidden Services*[4], to their respective entry nodes. In such attacks, the Tor client (and not the server) colludes with the

[3] A *circuit* is a temporary TLS connection between the anonymously communicating peers, involving the anonymization relays, through with the encrypted packets are transported.

[4] *Hidden services* are TCP/IP based services that use Tor to hide their true IP address or DNS name.

adversary and injects a traffic perturbation pattern that travels towards the Hidden Server. Such attacks are very similar to the efforts of Cheswick et al. [Burch and Cheswick, 2000] to traceback congestion-based network denial of service attacks.

Initially we evaluated our attack strategy in controlled environments, where we were able to accurately trace back to the source of the connection. Thereafter, we moved to experiments involving Tor clients communicating to a server, that we (the adversary) controlled, through public Tor relays. The client downloaded a large file from the server; the latter perturbed the bandwidth of the TCP connection, causing fluctuations that propagated across Tor all the way through the Exit Node to the client. Simultaneously, we tried to observe change in bandwidth on Tor relays and routers, connecting the anonymous client to the server, using LinkWidth. In these experiments we were able to identify approximately 49.5% of the routers connecting anonymous clients and servers, to their respective Tor entry nodes. We hypothesize that powerful adversaries, equipped with maps of the ASes, showing AS paths between the Tor relays and the potential clients, and several high bandwidth probing nodes, running LinkWidth, could use our attack to traceback the path between the clients and the Tor relay. In fact powerful adversaries, like the NSA, have been trying to build such maps [Risen, 2013].

The second attack relies on statistics obtained from network monitoring infrastructure installed in routers (*e.g.*, Cisco's NetFlow), to launch a practical traffic analysis attack, that involved correlating the statistics of the traffic flowing to and from the Tor network. To study this attack, we assumed an adversary that is somewhat similar to that assumed by Murdoch and Zieliński [Murdoch and Zieliński, 2007]. Such attackers have access to traffic flowing to or from a large number of relays. However, we were constrained by our capabilities. We had access to traffic flowing to and from a single Tor relay (that we hosted in our university), that could be operated only as a non-exit node. The server-to-exit data was gathered from the server, which we (the adversary) controlled. Alternately, one could assume that the adversary uses known traffic analysis attacks [Murdoch and Danezis, 2005; Mittal *et al.*, 2011] to identify the relays involved in an anonymous connection, and thus correlate traffic flowing to and from only these relays for identifying the source of an anonymous connection. Here again we assumed that the victim downloads a large file from

the server (for about 5 – 7 minutes), while the server (in collusion with the adversary) injects a traffic perturbation pattern in a connection that it sees arising from an exit node. The adversary then tries to correlate the deliberately injected traffic pattern to traffic statistics for connections that used the *entry node* (the first node in a Tor network path). The victim client's traffic statistics are expected to be most correlated to the injected pattern.

Similar to our experimental evaluation of the attack using our single-end controlled bandwidth estimation tool, we evaluated this attack first in an in-lab Tor testbed, and later involving data gathered for real Tor circuits. In the former experiments, we were able to identify the victim client with 100% accuracy. In experiments involving data gathered from real Tor circuits, we were able to correctly identify the victim in about 81.6% of the cases (with about 5.5% false positives). These results are explained further in Chapter 4.

There have been several proposals in the past to defend against traffic analysis attack that involve transmission of dummy traffic (also known as *link padding* [Shmatikov and Wang, 2006; Fu *et al.*, 2003b; Fu *et al.*, 2003a]). None have however been implemented due to fears of performance degradation. We present a way to defend against our NetFlow based traffic analysis attack, that involves transmission of dummy traffic consisting of packets with IP headers having small TTL values. These packets artificially distort NetFlow statistics, while having least impact on end-to-end performance. Although we employed such dummy traffic transmission schemes to defend against our NetFlow based traffic analysis attack, they could be effective in defending against other kinds of traffic analysis attacks as well. Initial experimental measurements revealed our approach to be beneficial in defending against our NetFlow-based traffic analysis attack.

Users of Tor continue to trust the system and expose sensitive information to the potentially untrusted relays. As mentioned above, an anonymous sender encrypts the data in multiple layers of encryption such that the anonymization nodes along the path between the sender and the receiver can neither inspect the actual data being transmitted nor have any way to know the identity of the actual communicating peers. However, the last node of Tor's anonymous path of cascaded proxies (known as an *exit node*), can observe the original data in plain-text. Malicious Tor exit node operators can eavesdrop on users' traffic and snoop out sensitive information such as usernames and passwords. In fact recent research

efforts [Winter and Lindskog, 2014] reveal that malicious exit nodes can not only observe plain-text traffic but could also launch SSL Man-in-the-Middle (MITM) attacks.

Networks like Tor are prone to Sybil attacks. In such attacks, an adversary runs few malicious nodes, advertizing fake bandwidth, attracting large number of clients, that he or she could de-anonymize using traffic analysis attacks. Powerful government organizations could operate nodes with high bandwidth in several networks to collect sensitive data and aid traffic analysis attacks. We have explored ways to identify such malicious exit nodes by deliberately exposing them with "sensitive appearing" but fake traffic (such as fake usernames and passwords for e-mail accounts), destined to a decoy server under our control. Periodically tallying the client and server logs for unsolicited login attempts, involving the decoy accounts, helps reveal malicious Tor exit nodes. Such eavesdroppers, having access to traffic entering and leaving exit nodes, could also be potential traffic analysis attackers. Therefore, by determining eavesdroppers, one could be potentially identifying traffic analysis attackers. In Chapter 6, we present a prototype system for our strategy involving decoy traffic for some common TCP/IP services that support plain-text user authentication messages. We also describe how our system could be adapted to support various kinds of protocols, including those relying on TLS sessions.

1.0.3 Problem Description and Hypothesis

As described above, it is generally feared that low-latency anonymity systems are vulnerable to traffic analysis attacks. However, it might be difficult, for even the most powerful adversaries capable of monitoring large number of networks and having tremendous computing power to correlate traffic patterns in different network segments, for launching traffic analysis attacks that unveil the identity of an anonymous connection. As is evident from previous research, de-anonymization could be conceived as a two-step process. The first step involves finding anonymization nodes, routers, ASes or IXes to monitor (depending upon the attack strategy and methodology), that likely transport the victim's traffic. The second step involves finding the victim or the source of an anonymous connection by observing the traffic flowing through this set of relays and routers. This primarily involves launching a form of traffic analysis attack on the network connections using the anonymization nodes or

routers, determined from the first step, and correlating traffic flowing into and out of these nodes or routers to associate otherwise unrelated connections (thus identifying the anonymous client). In the past, there have been adequate efforts to explore ways of finding the set of anonymization nodes or routers that very likely carry the victim's traffic. However, there have not been enough research efforts to explore the practical challenges involved in launching traffic analysis that rely solely on network statistics, and are aimed towards unveiling the identity of anonymously communicating parties. Moreover, the various methods to defend low-latency anonymization networks against traffic analysis attacks remain unexplored due to fears of performance degradation. Is it feasible to determine the source of anonymous traffic solely using network characteristics (*e.g.*, bandwidth and round trip time)? How accurate is it to determine the source of anonymous connection using such network characteristics? Can an attacker rely on existing network monitoring infrastructure to launch such attack? Can such traffic analysis attacks be thwarted without expending too much resources through methods that do not reduce the users quality of service? Can traffic analysis attacker be possibly detected? These are the kinds of questions which we are trying to answer in our research.

*Our research demonstrates that it is possible to confirm the source of anonymous traffic using methods that are not very invasive and rely solely on using network characteristics (*e.g.*, bandwidth), that could be observed through various less obvious means (*e.g.*, remotely estimating the available bandwidth of network links or simply by monitoring sampled traffic statistics available through monitoring infrastructure such as NetFlow). Moreover, it is also possible to defend against some of these attacks by using dummy traffic. Such traffic consists of packets with IP headers having small TTL values. These packets are dropped within a few network hops. They distort network statistics in a way that thwarts the process of identification of the victim, without affecting the users' performance. Other possible methods, utilizing Tor's built-in traffic shaping and conditioning parameters are not successful in defending against such traffic analysis attacks.*

A powerful large scale adversary, having the capability of inspecting traffic in various network locations (*e.g.*, Autonomous Systems or Internet Exchanges), through which large fractions of Tor traffic may transit (such as the adversary conceived by Johnson et al.

[Johnson et al., 2013]), might use such techniques to launch active traffic analysis attack to de-anonymize anonymous clients and servers.

Such adversaries, capable of launching traffic analysis attack by observing traffic transiting the Tor network, could potentially eavesdrops on the traffic between the exit node and the server. Thus, one way to determine potential traffic analysis attackers is to determine malicious adversaries who eavesdrop on traffic flowing out of a Tor exit node. Therefore, in addition to the traffic analysis attacks and mechanism to defend against them, we also propose a strategy to detect malicious eavesdropping exit nodes. Our strategy relies on exposing plain-text decoy information, unique to each exit-node, destined to a decoy server under our control and periodically tallying the client and server logs for unsolicited connection attempts information containing the previously exposed decoy information (such as fake user credentials). Such unsolicited connection attempts, arriving with some previously exposed decoy information, unique to an exit node, help identify the node that potentially eavesdrops on users traffic.

1.0.4 Important Assumptions and Attacker Models

Our attacks primarily focus on identifying the source and (or) destination of individual anonymous connections, among several competing connections. Even for the most powerful adversary, a brute force search involving the comparison of network statistics in each and every network link, could incur exponential costs. The de-anonymization process could thus be conceptualized broadly as a two-step process. The first step involves reducing the set of relays or routers to be monitored. The second step involves identifying the victim within this reduced set of relays or routers.

Our attacks specifically focus on the second stage of the attack. The first attack involves verifying the identity of the relays and routers involved in an anonymous communication session. The adversary tries to correlate changes in available bandwidth of an anonymous communication session, to variation in available bandwidth of the possible relays and routers connecting the relays to the possible clients. We assumed that a powerful adversary, in control of several probing nodes with high speed access to the Internet , correlates changes in available network bandwidth of relays and routers and tries to link those using,

LinkWidth. This process is prone to inaccuracies, primarily due to inadequate bandwidth and background congestion. To increase the accuracy of identifying relays and routers involved in an anonymous connection, we assume that the attacker controls the server so as to inject a specific network bandwidth perturbation in the victim's anonymous connection. These perturbations manifest as available bandwidth variations in relays and routers involved in an anonymous connection. The adversary tries to detect these perturbations using LinkWidth. As discussed above, the server may host content that may be of interest to potential Tor users or might simply host decoy multimedia files, and could wait for the users to connect to them. Alternately, the server could inject fake traffic along with regular HTTP responses, simply to congest the network links for a relatively long duration and aid the attacker to perform the measurements. It is assumed that a powerful adversary, equipped with maps of ASes, presenting AS paths between the ASes that hosts the Tor relays and those that host the potential clients, could traceback the path from the relays to the clients, using LinkWidth. However, it is not essential for the adversary to control one end of the anonymous communication. The adversary could use other means to deliberately perturb the network traffic without the server knowing about it.

The second attack assumes an adversary, which is either powerful enough to observe traffic flowing to and from a large number of relays [Murdoch and Zieliński, 2007], or uses attacks such as [Murdoch and Danezis, 2005; Chakravarty *et al.*, 2008b; Mittal *et al.*, 2011] to directly identify the relays involved in an anonymous communication. The adversary observes traffic statistics for the traffic flowing through these relays from existing infrastructure, such as Cisco's NetFlow, installed in most Cisco routers. The objective of the adversary is to correlate statistics for traffic entering and leaving the Tor relays participating in an anonymous communication, so as to identify the source of an anonymous communication. The adversary attempts to correlate the statistics for a specific anonymous communication to those corresponding to connections using the Tor entry node, to identify the anonymous client. Here again we assume that the adversary controls the server, so as to inject a traffic fingerprinting pattern which makes it easier to identify the network connection that corresponds to the source of the anonymous connection.

Finally, in Chapter 6 we present our strategy to detect eavesdropping by malicious Tor

exit nodes. As mentioned above, just about anyone could launch his (or her) exit node. A traffic analysis adversary may launch several relays (both entry and exit relays), with high bandwidth to attract traffic that it could de-anonymize using traffic analysis attacks. The adversary may also snoop on users' traffic to gather information about them. Our strategy can be used to detect such eavesdropping exit nodes (some of which could potentially be party to traffic analysis attacks). We present a proof of concept of our system to demonstrate how it could be used to identify malicious Tor exit nodes. In Chapter 6, we describe how the system can be extended to detect eavesdropping using different kinds of protocols.

Conducted from a purely academic standpoint, our research explores the technical challenges in de-anonymization of anonymous traffic, a way to detect the potential attackers and defend against them. Our research has been conducted primarily to adjudge the real threat to such systems and ways to defend against them, albeit from the point of view of a highly resource-constrained adversary. Presently, our attacks are targeted towards specific Tor users (*e.g.*, those that connect to our server) and can be used to de-anonymize one connection at a time. In future, powerful adversaries, equipped with adequate monitoring and probing nodes, could potentially use our attacks to simultaneously de-anonymize several anonymous connections. For each connection, the adversary could inject a unique traffic perturbation pattern and observe the pattern manifest on relays and routers involved in the respective connections. Moreover, the adversary could reduce the number of network locations to monitor by identifying key autonomous systems that transport traffic flowing to or from relays that potentially transport large fraction of Tor traffic (*e.g.*, relays advertising large bandwidth). We plan to explore these scenarios in our future research.

Thesis contributions: The main contributions of our research efforts may be summarized as follows:

- We present two *active* traffic analysis techniques, involving sparse information, to lead an adversary towards the source of an anonymous connection. The first attack involves the verification of Tor relays and network routes along the path, using a novel bandwidth estimation tool (called LinkWidth). The second attack involves correlating sampled network statistics, for traffic flowing into and out of Tor relays available

from existing networking traffic (*e.g.*, Cisco's NetFlow), to identify the source of an anonymous connection. Both these attacks are novel steps towards exploring traffic analysis attacks (involving network statistics) to identify the source of an anonymous connection.

- We also present a novel technique to defend against the NetFlow-based traffic analysis method that involves transmission of dummy traffic. Our experimental evaluation of this technique revealed that it is successful in defending against the attack without degrading users' quality of service. We also explored using some of Tor's built-in traffic shaping and conditioning features to thwart such traffic analysis attacks. Our experimental studies revealed that these features cannot be used to defend against such attacks.

- Lastly, we present a strategy to detect traffic eavesdropping by malicious Tor exit nodes. Our system regularly transmits unique decoy traffic to decoy servers that we control via the exit nodes. Periodically, the client and server logs, are tallied to identify unsolicited connection attempts on the server. Over a period of thirty-two months the system detected 18 different eavesdropping incidents. These malicious exit nodes *could be involved* in traffic analysis attacks.

Thesis roadmap: The next chapter of this thesis presents the reader with the necessary background information related to the Tor anonymous communication system, various attacks that have been explored in the past, proposed defenses against traffic analysis attacks and research generally related to eavesdrop detection in various networks, such as Tor. The third chapter presents our novel traffic analysis attack that involves use of our single end-controlled remote bandwidth estimation tool. The fourth chapter presents the traffic analysis attack involving statistics obtained from NetFlow data. In the next chapter we describe the efforts we undertook to defend against our NetFlow attack. The sixth chapter presents our strategy for eavesdrop detection in anonymization networks. Lastly, in our final chapter, we conclude the thesis with an overall discussion on the findings, limitations and future work.

Chapter 2

Background Information and Related Research

2.1 Background Information

2.1.1 Anonymous Network Communication Systems

Anonymous network communication systems enable users to hide their network identity (e.g. IP address) from their communication peers and also prevents network eavesdroppers to know the actual source or destination of messages. Most of these systems rely on sending traffic via one or more proxies, and may additionally encrypt traffic [Chaum, 1981] to obfuscate the true source or destination of messages (as described ahead in detail). Such systems are often classified as *low-latency* and *high-latency* anonymous communication systems. Low-latency systems are designed to be efficient for *semi*-interactive applications such as web browsing and instant messaging. High-latency systems are geared towards delay tolerant applications such as e-mail. Low-latency network anonymization systems are further classified based on the routing paradigms they employ–those that are derived from Onion Routing [Dingledine *et al.*,], and those that are based upon Crowds [Reiter and Rubin, 1998]. Systems such as Tor [Dingledine *et al.*, 2004], JAP [JAP,], and I2P [i2p,] employ deterministic routing, wherein the set of proxies through which the traffic is sent is known by the connection or session initiator. Systems such as GNUNet [Bennett and

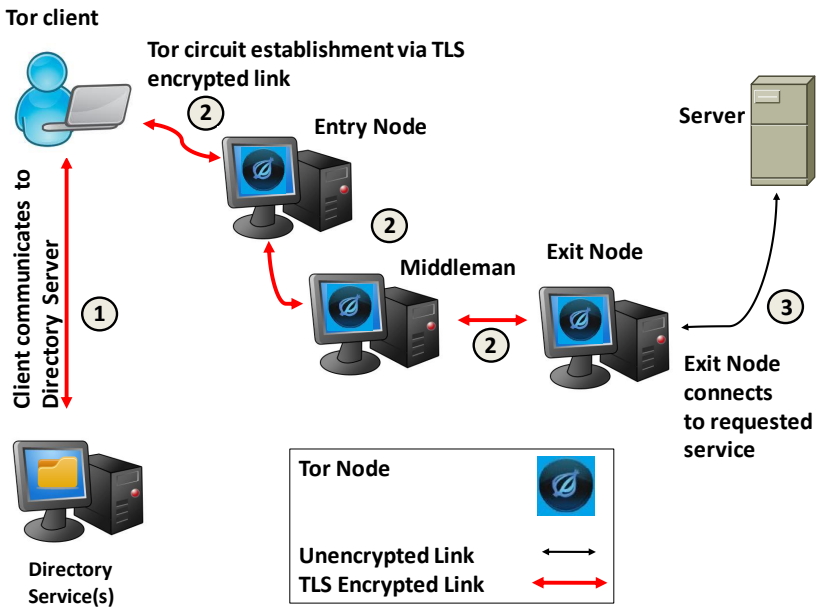


Figure 2.1: Basic steps for communicating through Tor. The client obtains a list of the available Tor relays from a directory service ①, establishes a circuit using multiple Tor nodes ②, and then starts forwarding its traffic through the newly created circuit ③.

Grothoff,], BitBlender [Bauer *et al.*, 2008], and OneSwarm [Isdals *et al.*, 2010] employ probabilistic traffic routing schemes similar to Crowds. Each traffic forwarding relay in such a system randomly chooses to send the traffic either to the destination or to another relay in the system.

Tor

Tor [Dingledine *et al.*, 2004] is closely modeled on the Onion Routing [Reed *et al.*, 1998] paradigm, and is one of the most widely used low latency anonymity networks, with an estimated user base of more than 500,000 users (as of May 2013 [Tor Metrics Portal,]). Tor aims to protect the anonymity of Internet users by relaying user TCP streams through a network of overlay nodes run by volunteers. The Tor overlay network consists of over 2500 proxies, known as *Onion Routers* (ORs), which are mostly operated by volunteers scattered across the globe. User traffic is relayed through *circuits*, which are formed by persistent

TLS connections between different nodes. By default, Tor circuits consist of three nodes: the first one is known as the *entry node* or *guard node*, the second one as the *middleman*, and the third is called the *exit node*. During circuit establishment, a Tor client negotiates shared secret keys with the relays that it chooses for the circuit. Thereafter, the client uses these keys to encrypt transmitted messages in multiple layers of encryption, starting with the key it shares with the exit node. Each of the nodes then first "peels off" one layer of encryption and then forwards the message to the next node on the circuit. The exit node decrypts the final layer of encryption, which reveals the original plain-text message of the user, and forwards it to its actual destination through a regular TCP connection. Thus, if in-transit traffic is intercepted by eavesdroppers, they cannot determine the actual source and destination of the traffic. This encryption scheme is schematically through Figure 2.2.

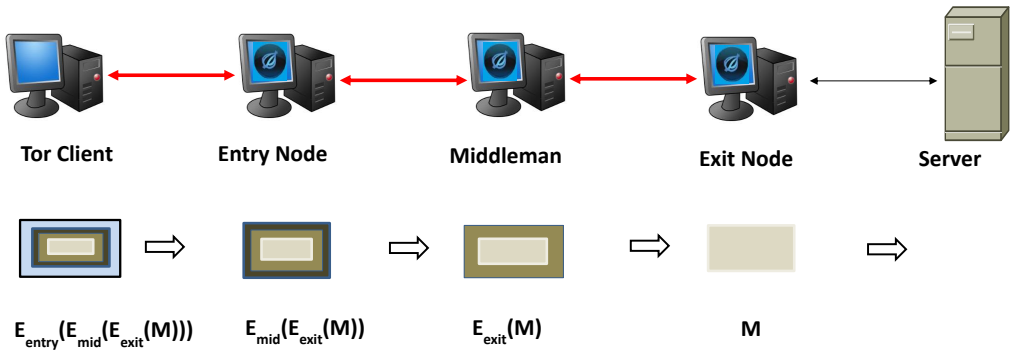


Figure 2.2: Layered (telescopic) encryption used by Tor client to encrypt Tor cells. Each relay decrypts one layer of encryption and forwards the resultant cells to the next relay along the path. The exit node sees the original packet (M) and finally transmits it to the server.

Figure 2.1 presents the basic steps for the creation of a new Tor circuit consisting of three onion routers:

1. The Tor client queries the directory service to obtain a list of the available Tor relays.

2. Then it establishes TLS connections to a node that it selects as the entry node. The entry node may already have established TLS connections to other nodes, which act as middlemen node. Those nodes again may already have connections to nodes that

may act as exit nodes. If there are no TLS connections between the entry node and the middlemen and middlemen and exit node, then those connections are established.

3. The Tor client thereafter establishes Tor circuits using the circuits, through the TLS connections (established in the previous step). The process involves, amongst other activities, the establishment of shared secrets with the relays.

4. The client then selects one of the circuits and establishes the TCP connection to its communication peer (the server), through this circuit.

Tor also provides configurable access control features to exit node operators. Usually, Tor exit nodes are configured to allow traffic forwarding for only a small set of TCP services. The supported services are defined by the operator of the exit node through the specification of an *exit policy*.

Hidden Services : Tor supports responder anonymity through Hidden Services. Responder anonymity allows a server to provide a TCP service without revealing its IP address. Hidden Services prevent against attacks that require IP address of the server. In this section we present an overview of how Hidden Service work.

Generally, service URI to IP address translation is done using the Domain Name System (DNS). For a Hidden Service, the regular DNS name used within the TCP/IP model, is replaced by a pseudo-random string (derived from the long-term public key of the server) ending with ".onion" domain name. A query to resolve a service URI ending in ".onion", can be resolved only within the Tor network. This new URI and the long-term public-key, representing the service, is published by the server, the first time it joins the Tor network. Only a Tor user can thereby access the service through an anonymous Tor circuit connecting himself to Hidden Service. This is shown in figure 2.3. As shown in the figure, the client and server establish connections to relays called *Rendezvous Point (RP)* and *Introduction Points*, respectively. A form of Diffie-Hellman key exchange ensues between RP and one of the Introduction Points. As an outcome of this exchange, the client and server circuits are connected with one another and communication between the client and server ensues. A more detailed description of this handshake procedure is presented in the Tor design document [Dingledine *et al.*, 2004].

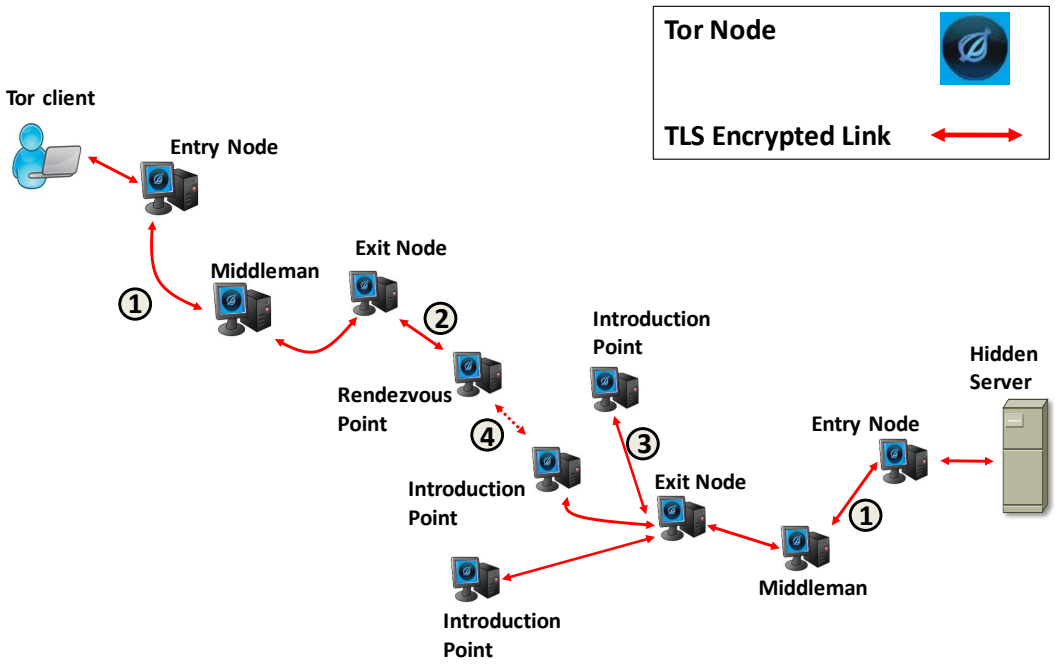


Figure 2.3: Basic steps for communicating to hidden services. The client and server create Tor circuits① to nodes called *Rendezvous Point*② and *Introduction Points*③, respectively. The client's query to resolve ".onion" URI gets returned with information of the Introduction Points. Finally the Rendezvous Points connect to Introduction Points and thus communicates to the hidden server④.

2.2 Related Research

2.2.1 Attacks Against Anonymous Communication Systems

In general, de-anonymization of anonymous communication is a two-step process. The first step involves finding the *anonymity set*. In anonymity and privacy parlance, the anonymity set is the set consisting of anonymized entities (e.g. computers, humans, etc.), whose true identities have been hidden using some anonymization scheme. The observer of such a set observes the actual set and the set containing the former's anonymized identities, but has no way to determine the relationship between the two sets. From the point of view of

attacks against anonymous communication networks, involving identification of a certain anonymous client, every host or network on the Internet, could be a potential victim. However, it is not feasible, for even powerful adversaries, to monitor each and every network or host, for identifying the source of anonymous traffic. Various research efforts have been thus carried out in the past to reduce the set of hosts or network routers to monitor. For the sake of convenient description, in this thesis, we call this reduced set as the anonymity set. In practice, some researchers have described how this anonymity set could be determined through a Sybil attack, wherein the adversary runs several malicious relays with the hope that some of these would be selected in users' connections and would aid an adversary to observe traffic entering and leaving the anonymization network. Other strategies, assuming powerful adversaries, involve observing traffic entering and leaving the the network by observing traffic in vantage Autonomous Systems (ASes) or Internet Exchange Points (IXes), intervening the paths from various networks to the relays of anonymization networks [Murdoch and Zieliński, 2007; Edman and Syverson, 2009b; Johnson *et al.*, 2013; Feamster and Dingledine, 2004].

The second part of the de-anonymization process deals with finding the actual source of anonymous traffic by monitoring network connections that use the routers or network relays, that make up the anonymity set, and transport the victim traffic This generally involves some form of traffic analysis attack, wherein the adversary, having access to traffic in various networks, can correlate traffic transiting the anonymization relays with traffic flowing to (or from) the anonymity set, and identify the source of anonymous traffic. Low-latency anonymous communication systems, geared towards semi-realtime applications, try to assure users' quality of service, by not modifying packet inter-arrival characteristics, such as delay and jitter. This makes them particularly vulnerable to traffic analysis attacks [Wright *et al.*, 2002; Shmatikov and Wang, 2006].

Step 1: Finding the anonymity set

Researchers in the past decade have explored various methods to determine the anonymity set. Some of these efforts included Sybil attacks, wherein the adversary runs malicious Tor entry and exit relays, with the hope that they would get selected in circuits, and the

node operators would be in a position to observe traffic entering and leaving the Tor network. Such an attack was first explored in 2007 [Bauer *et al.*, 2007], wherein the authors proposed attracting large fraction of Tor traffic by running malicious relays that advertised high available bandwidth. An anonymous client could select such malicious relays in entry and exit positions, which might be engaging in traffic analysis attack. Such attacks succeed because Tor relay selection is biased on advertised bandwidth [Tor Path Specifications,].

In a related attack, Pappas et al. [Pappas *et al.*, 2008] suggested creating looping circuits across non-malicious Tor relays and keeping relays busy, so as to prevent them from being selected in circuits. In the meanwhile the adversary could run malicious relays, that might advertise high bandwidth to increase their relative chance of being selected in circuits, especially while the benign nodes are busy serving malicious circuits specifically designed to keep them busy.

As mentioned previously in the Introduction, Øverlier et al. [Øverlier and Syverson, 2006], proposed building of Tor paths, solely using nodes, known as *guard* nodes, so as to avoid such attacks. The threat of such Sybil attacks, involving malicious relays attracting users' traffic, is however only partially mitigated. Malicious relay operators could deploy nodes with high bandwidth for a certain period of time, so as to gain adequate trust, before launching in traffic analysis attacks.

Feamster el al. [Feamster and Dingledine, 2004], and later Edman el al. [Edman and Syverson, 2009b], having studied the topology of the Internet concluded that on an average, in 22% of Tor circuits originating from different subnets, there are ASes which can observe traffic going towards a Tor entry node (from various subnets) and from exit nodes to some popular destinations (e.g. popular search engines and free web-mail services). More recently, Johnson et al. [Johnson *et al.*, 2013] showed that a small number of compromised Tor relays that advertise high bandwidth and IXes observing both entry end exit traffic, can de-anonymize 80% of various types of Tor circuits within about six months.

In 2007, Murdoch and Zieliński [Murdoch and Zieliński, 2007], showed primarily through simulations, that a small set of IXes, could observe a traffic entering and leaving the several Tor entry and exit nodes within UK. They further showed, through simulations, that NetFlow [Claise,], a traffic monitoring system installed in commodity routers, could

be used to launch analysis attacks against Tor. The results were however mostly based on simulations, involving data obtained from by observing a single Tor relay. The research does not provide any intuition of the accuracy one can expect while practically using NetFlow based statistics to de-anonymize anonymous traffic. *As described ahead, our research attempts to fill this gap by performing active traffic analysis to de-anonymize Tor clients and exploring the accuracy and practical issues involved in de-anonymizing anonymous traffic using NetFlow data.*

Traffic analysis techniques have been explored in the past, for determining the anonymity set. In 2005, Murdoch and Danezis [Murdoch and Danezis, 2005] developed the first practical traffic analysis attack against Tor. They proposed a technique to determine the Tor relays involved in a circuit. The method involved a corrupt server, accessed by the victim client, and corrupt Tor client that could form one-hop circuits with arbitrary Tor nodes. The server modulates the data being sent back to the client, while the corrupt Tor node is used to measure delay between itself and Tor nodes. The inverse correlation between the perturbations in the client server traffic, deliberately introduced by the corrupt server, and the one way delay, measured by the corrupt Tor client helped identify the relays involved in a particular circuit.

In 2009, it was demonstrated by Evans el al [Evans *et al.*, 2009] that the traffic analysis attack proposed by Murdoch and Danezis was no more applicable due to the large number of Tor relays, the large volume of Tor traffic, low end-to-end quality of service and possible network bottleneck locations between the adversaries' vantage point and the victim relays. They proposed a method to amplify the network traffic by using circuits that repeatedly used the same relays and aided in easier identification of the relays.

We proposed a traffic analysis method using remote network bandwidth estimation tools, to identify the Tor relays involved in Tor circuits [Chakravarty *et al.*, 2008b]. Our method assumed that the adversary is in a position to perturb the victim traffic by colluding with the server and is in control of various network vantage points, from where he can remotely observe variations in network bandwidth. We were able to confirm the identity of the relays involved with a success rate of 59.46%. In our experiments, we used our remote bandwidth estimation tool, called Linkwidth (described ahead in Chapter 3).

In 2011, Mittal *el al.* [Mittal *et al.*, 2011], demonstrated a somewhat modified version of the Murdoch and Danezis attack, which rather than using variation in one-way delay, relied on bandwidth variations to determine if two clients were using the same set of Tor relays for their circuits.

Step 2: Identifying the source of anonymous traffic within the anonymity set

As described previously, there have not been adequate studies to explore methods to identify the source of anonymous traffic within the anonymity set, that solely rely on network statistics. Amongst the very few previous research efforts, Hopper et al. [Hopper *et al.*, 2007; Hopper *et al.*, 2010] presented comprehensive efforts, using an information theoretic approach, to identify the Tor relays and thus eventually the possible location of the Tor client. In their efforts ,they combined the technique presented by Murdoch and Danezis', along with one-way circuit latency, and the Vivaldi network coordinate system to determine the possible source of anonymous traffic. With moderate accuracy (median information loss of 2.46 bits and less than 1 bit information loss for 23% of trials), they there were able to estimate the location of the client.

Traffic analysis attacks can be considered a "off-shoot" of *side-channel* [sidechan,] attacks wherein the adversary relies on the correlation of change in anonymous network traffic statistics and changes in various related system parameters such networking equipments' memory usage and temperature. For example, in a previous work, Murdoch et al. [Murdoch, 2006; Zander and Murdoch, 2008] demonstrated how variation in anonymous traffic leads to variation in CPU temperature and, hence, the system clock. TCP clock skew, which can be remotely determined through TCP timestamp options, could be used to reveal this system clock drift. Thus changes in various network and system parameters, such as throughput and one-way (or round trip) latency, that can lead to correlated variations in temperature of network equipment can be determined through TCP timestamps, and thus aid in identifying the source of anonymous traffic.

In Chapter 3, we present a Traffic Analysis attack for confirming the identity of anonymous client, using our remote bandwidth estimation method, through a kind of traceback method. This is an extension of our previous work to use remote bandwidth estimation

method to identify relays involved in a Tor circuit. In our research, we assume that the adversary is powerful, is endowed with high bandwidth nodes in vantage locations, has a map of the Internet showing the connectivity of routers or ASes and colludes with a malicious server (that is visited by anonymous users). The last assumption allows him (or her) to force the server to inject a traffic perturbation pattern that travels to the victim client, via Tor nodes and intervening relays. Our method may seem somewhat similar to the attack presented by Murdoch and Danezis [Murdoch and Danezis, 2005] and Mittal et al. [Mittal *et al.*, 2011]. However, unlike their method, ours is not restricted to confirming the relays participating in a Tor circuit. We used our method to launch a form of traceback method to confirm the routers connecting the entry node to the client.

This attacker model has been carried forward in our research to explore the practical feasibility of traffic analysis attacks using network monitoring tools installed in existing networking infrastructure, e.g. Cisco's NetFlow. In our research we explored the capabilities of an adversary, that uses sparse NetFlow records, corresponding to the traffic entering and leaving the Tor network, to identify the client whose traffic characteristics were most correlated to the server injected traffic perturbation pattern. Our experiments were carried out in two stages. The first was carried out in an in-lab set-up using a testbed, free from external network congestion and disturbances. In these experiments, we were able to identify the victim in 100% of the cases. The second stage of experiments were carried out in using data obtained from a real Tor relay. In these experiments, we were able to correctly identify the victim client in about 81.4% cases (with about 6.4% false positives).

In their research [Hopper *et al.*, 2007; Hopper *et al.*, 2010], Hopper et al. have presented ways to determine if two circuits use the same relays, identity of the relays involved in a circuit and finally identification an anonymous client. Their approach however assumes that two network connections might be originating from the same source network if the end-to-end delays of the Tor paths, connecting the clients to the server, are of comparable orders of magnitude. One cannot say if such assumptions are still completely valid tn today's Tor network consisting of over 2500 relays that serve over half a million users. In contrast, our NetFlow based traffic analysis attack involves direct observation of the correlation between server-to-exit and entry-to-victim client traffic to identify the victim (the one that is most

correlated to the server-to-exit traffic statistics).

2.2.2 Defending Against Traffic Analysis Attacks

Most forms of proposed defense against traffic analysis attack involves sending dummy traffic (also often known as *link padding*) [Shmatikov and Wang, 2006; Fu *et al.*, 2003a; Wang *et al.*, 2008], deliberately dropping some packets, known as defensive dropping [Levine *et al.*, 2004], and injecting deliberate artificial delays. High latency mix based systems can tolerate performance degradation due to such artificial delays, low-latency anonymity networks cannot. Various researchers in the past have proposed transmitting dummy traffic to hide traffic patterns which could lead to traffic analysis attacks in low-latency anonymity networks. Some such as [Shmatikov and Wang, 2006; Fu *et al.*, 2003a; Wang *et al.*, 2008] propose strategies to send dummy traffic without resulting in considerable performance degradation. In 2003, Fu et al. [Fu *et al.*, 2003a] present analytical models to study effectiveness of traffic analysis countermeasures. Their model assumes a hypothetical adversary that primarily uses various metrics such as traffic rate and inter-packet arrival times to correlate otherwise apparently unrelated traffic flows. They assume a defense mechanism that involves sending dummy traffic co-ordinated with a programmable system interval timer. Their model was however only evaluated on a small controlled in-lab set-up.

In 2006, Shmatikov and Wang [Shmatikov and Wang, 2006] proposed an end-to-end dummy traffic sending scheme which they called *Adaptive Link Padding*. Their scheme involved observing the delays between packets "on-the-fly" and estimating the delay between packets that are yet to be seen, to determine if the gaps were natural or deliberately injected, so as to aid traffic analysis attacks. Accordingly the system would generate dummy traffic. They juxtaposed their Adaptive Link Padding scheme against constant rate dummy traffic transmission (where the the system transmits dummy traffic at a fixed rate) and showed how their efforts consumed lesser bandwidth when compared to the latter and was more effective in preventing traffic analysis attacks.

Wang et al. [Wang *et al.*, 2008] proposed a dummy traffic sending scheme, which they called *Dependent Link Padding*. The idea is somewhat similar to Adaptive Link Padding scheme proposed previously by Shmatikov et al. Rather than observing packet

inter-arrival delays, Wang et al. chose to observe incoming traffic rate to decide if, and how much, dummy traffic should be generated. The compare their scheme against constant rate padding, wherein it is assumed that dummy traffic is always sent at a constant rate without adapting to the bandwidth of existing traffic.

However, all such strategies are yet to be adopted by low-latency anonymity networks such as Tor. The main concern which inhibits widespread adoption is the effect on performance. Our dummy traffic sending mechanism is directed specifically to defend against NetFlow traffic analysis attack described in Chapter 4. Our motivation is to artificially distort NetFlow statistics, without significantly degrading clients' perceived quality of service. Our method relies on the entry node sending dummy traffic, which although have IP header and packet lengths identical to original Tor traffic, have very small TTL values, that cause them to be dropped within few network hops, close to the entry node. Such a scheme has very little impact on performance. Our method could be adopted to any form of traffic sending scheme. Our method is described in detail in Chapter 5.

2.2.3 Detecting Eavesdropping by Tor Exit Nodes: Detection of Potential Traffic Analysis Attackers

We mentioned in the beginning, that an adversary that eavesdrops on anonymous users' traffic, flowing out of exit nodes, is also a potential traffic analysis attacker. Such an adversary, observing traffic entering and leaving the Tor network, can not only correlate traffic patterns, but also directly observe the traffic between the exit node and the server, misusing the contents of traffic. Thus, one way to identify adversaries of anonymous communication systems, like Tor, is to identify malicious exit node operators that eavesdrop on traffic transiting through the exit node.

Our efforts to identify eavesdroppers of Tor traffic, is closely related to research efforts that involve the exposure of enticing decoy information or resources to lure potential adversaries, with the objective of identifying them and determining their modus operandi. One of the first uses of decoy information for enabling the observation of real malicious activity has been documented by Clifford Stoll [Stoll, 1988]. In his book, the author recounts his efforts to trap an intruder that broke into the systems of the Lawrence Berkeley National

Laboratory. As part of his efforts to monitor the actions and trace the intruder's origin, he generated fake documents containing supposedly classified information that would lure the intruder to come back and stay longer on the compromised computer.

Computer based systems and resources deployed widely with the objective of luring prospective adversaries and intruders for logging their identities and actions, are widely known as *honeypots* [Provos, 2004; Spitzner, 2003]. Such systems have no production value other than being compromised, and subsequently aid in tracking the actions of the attacker. Honeypots have been extensively used for modeling, logging, and analyzing attacks originating from sources not only external to an network [Honeynet, ; Yuill *et al.*, 2004], but also from within its perimeter [Bowen *et al.*, 2009b].

Complementary to honeypots, researchers and system administrators often use *honeytokens* [Spitzner,] which are pieces of information with no other purpose than being intercepted or stolen and abused by an adversary. Any use of these honeytokens clearly indicates unauthorized access. The decoy credentials used in our approach, which we describe in detail in the next section, can be classified as a variety of honeytokens.

More recently, Bowen et al. [Bowen *et al.*, 2009a] proposed the use of decoy documents to detect misbehaving entities within the perimeter of an organization. The decoy documents contained embedded "beacons," such as scripts or macros, which are executed when the document is opened. The authors used fake tax records bearing information appearing to be "sensitive" and enticing to an adversary. In case a document is opened, the embedded beacon connects to an external host and transmits information such as time of access and IP address of the hosts used to open the document.

In another related research work, Bowen et al. [Bowen *et al.*, 2010] used real WiFi traffic as a basis for the generation of decoy traffic with realistic network interactions. An API is used to insert bait content, such as popular webmail service cookies, FTP and HTTP protocol messages, and so on into these decoy packets. The packets were then broadcasted through an unencrypted WiFi network and exposed to potential eavesdroppers. Unsolicited connection attempts to the services, using the bait credentials, are marked as illegitimate. In their experiments, the authors replayed `gmail.com` and `PayPal.com` messages carrying credentials and cookies for decoy accounts, and utilized the last login IP address feature of

these services for determining illegitimate connection attempts. Such techniques are not applicable anymore as the aforementioned popular web mail and financial services encrypt their connections with SSL.

There has been little effort in detecting misbehaving overlay nodes of anonymity networks. In a work most closely related to ours, McCoy et al. [Mccoy *et al.*, 2008] attempted to detect eavesdropping on malicious Tor exit routers by taking advantage of the IP address resolution functionality of network traffic capturing tools. Packet sniffing tools such as `tcpdump` [McCanne *et al.*,], are by default configured to resolve the IP addresses of the captured packets to their respective DNS names. Their system transmitted, via Tor exit nodes, TCP SYN packets destined to unused IP addresses in a subnet owned by the system's operator. When the packet capturing program attempted to resolve the IP address of a probe packet, it issued a DNS request to the authoritative DNS server. The system's operator had access to the traffic going to this authoritative DNS server. Thus, requests to this DNS server with the unused IP address were an indication that probe packets had been intercepted by some packet capturing program, and could be traced back to the network host where they were captured. However, when capturing traffic on disk, `tcpdump` by default does not resolve any addresses; and in any case the eavesdropper can trivially disable this functionality, rendering the above technique ineffective.

In contrast to that work, our system does not require access to DNS traffic. We employ decoy clients and servers which communicate via Tor exits and present easily reusable sensitive appearing information such as user credentials and URLs to sensitive appearing documents which also contain beacons such as those described above. The system periodically monitors the client and server logs for unsolicited connection attempts to the server which are marked as suspicious. Our system, with various components has been extensively described in Chapter 6.

Chapter 3

Traffic Analysis Against Anonymity Networks Using Remote Bandwidth Estimation

3.1 Overview

In this chapter, we introduce our first novel, remotely-mounted traffic analysis attack that can be used to confirm the network identity of anonymization proxies (Tor relays), network routers, and hidden servers, involved in anonymous communication sessions. To help confirm the identity of an anonymous client, our adversary colludes with the server to deliberately inject traffic perturbations in an anonymous TCP connection, that the server sees arriving from an exit node. In this attack, the adversary, equipped with a novel remote bandwidth estimation tool, tries to correlate this deliberately injected traffic perturbations to available bandwidth variation in Tor relays and network routers involved in the circuit. The adversary employs our *single-end controlled* available bandwidth estimation tool (called LinkWidth [Chakravarty *et al.*, 2008c], derived from TCP Westwood [Gerla *et al.*, 2001] algorithm) to carry out this process.

Our traffic analysis attack approach is similar to the one proposed by Murdoch et al. [Murdoch and Danezis, 2005]. However, contrary to their approach , our technique

does not rely on modified Tor clients and can be used to track not only the anonymizing relays, but can also be used to launch a "trace-back" attack to the the victim's network location, through the network routers connecting the client to its entry node. We assume that the adversary controls, or can create a traffic fluctuation, in one end of the anonymous connection and can remotely detect it propagate through various relays and routers along the circuit leading to the client.

The attack involves a client downloading a relatively large file from a server, that perturbs the throughput of the TCP connection it sees originating from an exit node. One way to achieve this is through the use of traffic shaping and conditioning tools [Hubert *et al.*,]. The injected traffic perturbation travels through the circuit, via the relays and routers involved, to the victim client, since Tor does not alter traffic inter-arrival characteristics. An adversary, in a position to observe this perturbations remotely, can potentially track these perturbations to the client, through the relays and routers involved. In our research, we try to study the capabilities of such an adversary, who, equipped with LinkWidth and colluding with the server, so as to inject perturbations in traffic, can potentially verify the identity of the anonymous client by remotely observing these perturbations on relays and routers involved along the path intervening the client and the server.

To verify the validity of our approach, we performed a series of experiments. We evaluated the accuracy of our technique on controlled environments (using in-lab and DETER [DETER Network Security Testbed,] testbeds). In these situations, we achieved 100% success in verifying the relays and routers involved in victim circuits. Further, we tested our technique through a series of experiments involving the client establishing circuits via public Tor relays to servers that we controlled. In these experiments, we achieved varying success in exposing the identity of the victims. On an average, we detected 59.46% of the Tor relays participating in our circuits. In addition, we successfully identified the intermediate network routers connecting Tor clients and Hidden Services to their Entry Nodes in 49.5% of the cases.

We posit that a real adversary equipped with many appropriately positioned high-bandwidth hosts and a partial map of the network, could effectively attack timing-preserving anonymizing systems. Some prior efforts in generating such maps are the *Internet Mapping Project* [The

Internet Mapping Project,], *AS Peering Analysis* [CAIDA AS Peering Analysis,], *iplane* [Madhyastha *et al.*, 2006], and similar efforts by CAIDA [cai,]. Searching a map depicting the ingress and egress routers of an Autonomous System (AS) involves little complexity. An adversary equipped with such information would probe for the induced available bandwidth variation, *only on* the inter-domain routers in search for the AS that hosts the victim anonymous client[1]. The objective of our research however is not to demonstrate this search process but to answer the following question: *Can a well-provisioned (Semi-)Global Adversary equipped with probing nodes, scattered close to most inter-AS routers, and "sender-only" controlled bandwidth probing tools like LinkWidth, measure bandwidth fluctuation on network routers and anonymizing relays associated to a anonymous communication session?*

We begin this chapter with a description of our bandwidth measurement tool (LinkWidth). Thereafter, we describe the traffic analysis strategy. This is followed by a detailed description of our experimental evaluation of our attack strategy and the results obtained from the same. The chapter concludes with a brief description of the various issues related to our measurements strategy and limitations.

3.2 LinkWidth: Single End-Controlled Bandwidth Measurement Technique

We implemented a TCP Westwood [Gerla *et al.*, 2001] sender in LinkWidth, a tool for performing single-end network capacity and link throughput measurements. It requires no support or active collaboration from a remote host (or any device in the network). It primarily uses *packet train* [Prasad *et al.*, 2003] method for bandwidth measurements. Such bandwidth measurement methods, designed to measure the bandwidth between two networked hosts, relies on sending a long train of packets from one host to the other, so as to completely use up the end-to-end available network capacity for a short duration. LinkWidth is a single-end controlled bandwidth measurement tool that can be used to measure the end-to-end bandwidth between to hosts. A user can run it to measure the bandwidth of the

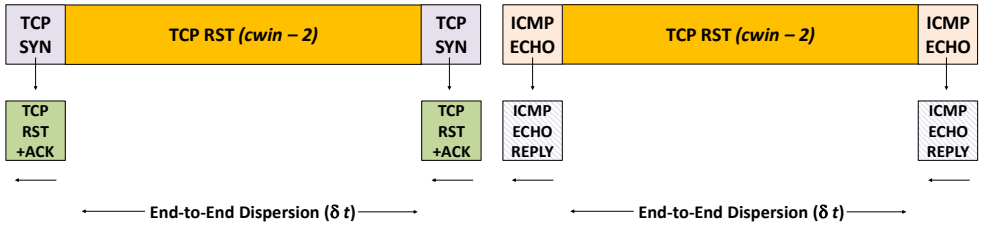
[1]Further, resolving down to the end hosts might require ISP router maps through services such as as *Rocketfuel* [Rocketfuel,]

path connecting his (or her) host and any other host, *without any support from the latter*. The probing host (sender), sends a train of TCP packets where the RST flag is set. This train is "sandwiched" between two TCP SYN packets destined to closed ports. The TCP SYN packets evoke TCP packets having RST and ACK flags set, as replies, back to the sender, while the intervening packets (having the RST flag set), are dropped by the recipient, without any response. The time difference between the TCP RST replies, often called the *dispersion*, is treated as the approximate time difference between the reception of the first and last TCP SYN packets, based on the assumption that the path between the sender and receiver is symmetric.

To measure end-to-end TCP capacity, the sender emulates the TCP Westwood sender by sending number of packets equal to the TCP congestion window ($cwin$). The packet train is arranged so that there are $cwin$ TCP RST packets (called *load packets*), are "sandwiched" between two TCP SYN packets (called the *head measurement packet* and *tail measurement packet*, respectively). Figure 3.1(a) shows this arrangement of packets. The sender emulates a TCP Westwood sender by initializing $cwin = 1$ (one TCP RST packet between two TCP SYN packets). Correct reception of the TCP packets with RST and ACK flag set, by the sender, signals that the train has been correctly delivered to the destination. The time difference between the received TCP packets with RST and ACK packets, called the dispersion, is used to determine the available bandwidth. If each of the packets is L bytes long, and if the dispersion of the responses is δt, then the path bandwidth is measured as $\frac{cwin*L*8}{\delta t}$ (measured in bits/s). Once the train is correctly received, the sender doubles the number of packets in the train, while measuring the bandwidth at each iteration. The process continues iteratively *ad infinitum*, exponentially increasing the number of packets in the packet train each time, while measuring the available bandwidth in each iteration, until the sender experiences packet losses, that is signaled by non-reception of replies, within a certain timeout interval[2]. At this point, the sender switches to congestion avoidance mode, computing the congestion window based on previously measured available bandwidth. The average of the available bandwidths, measured in the next few iterations, until another packet loss is

[2]Since we do not rely on a full-fledged TCP connection, the only way to signal a packet loss is by coarse timeout. After sending the train, the sender initiates a timer to wait for the two expected ACKs

experienced, is considered as the end-to-end available path bandwidth.



(a) Packet arrangement with TCP measurement packets

(b) Packet arrangement with ICMP measurement packets

Figure 3.1: (a)Arrangement of packets in LinkWidth for measuring available bandwidth, using TCP measurement packets, (b) and using ICMP measurement packets.

In some cases, the TCP SYN packets are filtered or rate-limited. To counter this, we replace the head and tail TCP SYN packets with ICMP ECHO packets. The load packets continue to be TCP RST packets. Correct reception of the train is indicated by reception of ICMP ECHO_REPLY packets at the sender. The arrangement of packets is shown in Figure 3.1(b). We developed a prototype for Linux. To avoid incurring packet delays due to kernel resource scheduling, we bypassed the regular protocol stack and send our own TCP and ICMP packets crafted using the Raw Socket API. The coarse timeout was implemented using the standard POSIX API function *setitimer()*. For a more detailed description of LinkWidth and how it emulates the TCP Westwood sender, we encourage the reader to read our technical report [Chakravarty *et al.*, 2008a].

3.3 Threat Model and Attack Approach

Before delving into the attack details, we discuss the threat model for which we claim that our attacks are effective.

Threat Model

Our primary focus is an adversary who can induce "traffic fluctuations" into a targeted anonymity preserving channel and observe it "trickle" towards the victim. The adversary may have hosts under his control on many ASes or could be at a network vantage point with respect to routers in various subnets. Each of these hosts may be running a copy of bandwidth estimation tools such as LinkWidth. They would also be at a network vantage point with respect to all candidate victim relays and network routers. This is feasible in toady's' Internet: the inter-router media connecting the routers of major tier-3 ISPs can support over 10 Gbit/s (let alone the tier-1 and tier-2 ISPs). Most have 30–40% under-utilized spare capacity. Therefore, while we are not in a position of having a large set of vantage points, this does not prevent others from being able to do so. Often, adversaries like us, might be sensing "under-utilized high-bandwidth" links. Popular anonymity preserving systems, like Tor relays, often dedicate a considerable fraction of the traffic to forwarding client traffic. However, such small fluctuations in high capacity links, are less than what most state-of-the-art available link bandwidth estimation techniques can detect accurately.

Traffic Analysis Methodology

We used LinkWidth to detect induced traffic fluctuations on candidate anonymizing relays. We identify probable candidate network routers that could reach this relay through a single network hop. Since most anonymity preserving relays are at the network edges, the network routers that could directly reach the candidate relays would be their default gateways. This intuition was re-applied on default gateways to determine which network interface, and hence the network routers within one network hop, exhibited similar fluctuations. We repeated the tracking process recursively until the fluctuations were tracked down to the source network (and possibly the source itself).

To quantify our detection capabilities, we performed extensive experiments initially in a controlled emulation environments, namely DETER [DETER Network Security Testbed,]. Some of the experiments were further validated in controlled lab-environment. We also uncovered real-world Tor clients, and hidden servers which communicated using public Tor relays, with some success. Our approach requires a "map" presenting with information of

inter-domain routers for the Tor Entry Node and the victim. In our experiments, we did not use elaborate maps. We only considered result obtained from running *traceroute* between the targeted victim and its Tor Entry Node. Moderate success rate is primarily due to a combination of our inadequate network vantage points and low end-to-end throughput offered by the Tor relays as compared to the available link capacities of routers and relays. Lastly, our accuracy was also affected by the presence of background cross traffic on regular network routers, resulting in higher false negatives when compared to the in-lab or DETERLAB experiments' results.

3.4 Experimental Evaluation

Our attack technique can be applied to low-latency anonymity systems that are based on *Onion Routing* [Goldschlag *et al.*, 1996]. Tor is a good example and chief among onion routing anonymizing based systems. In such systems, there seems to be a trade-off between anonymity guarantees and performance [Reardon and Goldberg, 2009]. We show that in both controlled and real-world experiments, a well-provisioned adversary can expose the anonymity of Tor relays and end-points. This is performed by determining the available bandwidth fluctuations on a large number of relays and network routers.

3.4.1 Experiments Using the DETER Testbed

To determine the effectiveness of our attack, we used DETER [DETER Network Security Testbed,] to emulate various network topologies. DETER is a network emulation system that offers commodity machines and switches connected with user specified topologies. Users can specify the operating systems, network connections, link speeds, routing protocols and other parameters.

Figure 3.2 depicts one of the topologies we used to validate our approach. In these experiments, we used Iperf [Tirumala *et al.*, 1997] and LinkWidth to detect the fluctuation of the available link bandwidth on network routers along the path connecting the server to the actual client[3].

[3]Due to the way DETER emulates the requested link capacities and the use of traffic shaping, Iperf was

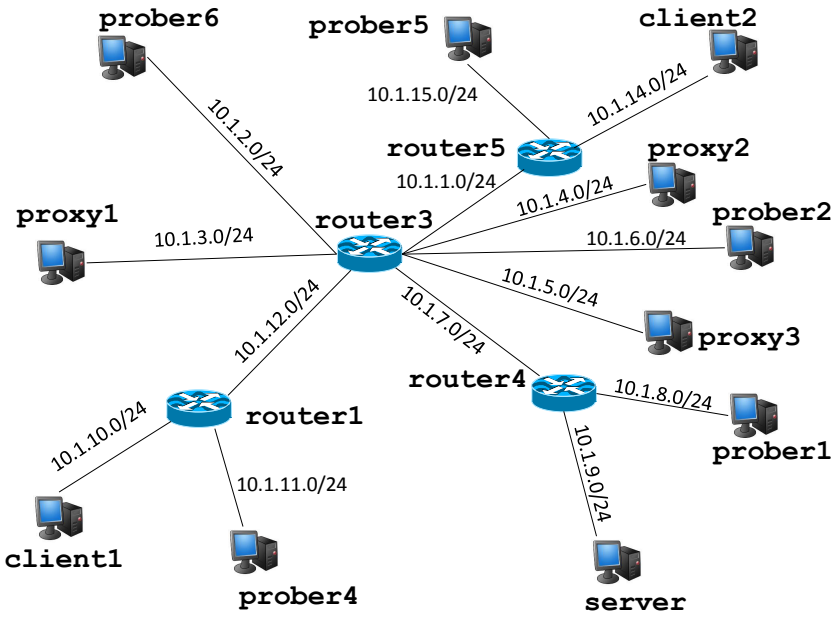


Figure 3.2: DETER testbed network topology.

The host, marked as `server` in the topology figure (Figure 3.2), was the colluding web server. In addition, there were two client hosts, `client1` and `client2`. `client1` was the victim downloaded a large file from `server`, using HTTP; `client2` was idle.

With two clients on two separate branches, the available traffic fluctuation induced by `server` was observable only by one of the branches (the one leading to `client1`). The probing hosts on each of the subnets probe the intermediate links. To forward packets through `proxy1`, we used `squid` [Wessels *et al.*,]. `client1` connected to `server` via `proxy1`. The `server` modulated the transmission rate to induce the necessary bandwidth fluctuation. The adversary probed for the bandwidth variation on the network elements in both branches using its probing hosts.

The reason we avoided installing an anonymizing system in this particular experiment is due to the poor QoS resulting from computational and network transport costs in systems like Tor. The motivation behind choosing `squid` was to demonstrate the effectiveness of

more accurate than LinkWidth in estimating the bandwidth variations

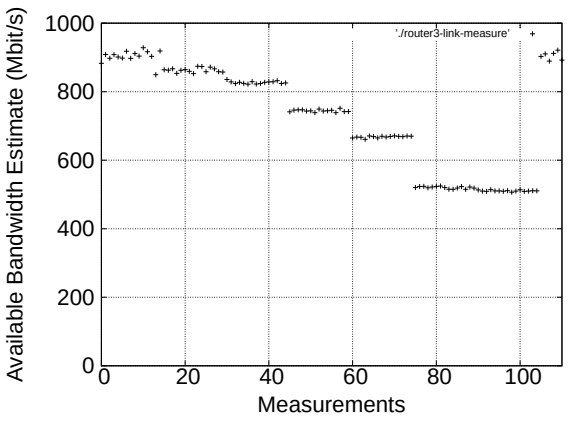


Figure 3.3: The detected available bandwidth on the router connected to the victim `router3` drops uniformly as client traffic increases.

our hypothesis; we assume that the relaying architecture of Onion Routing based systems would soon provide higher throughput rate like unencrypted `squid` proxy.

Therefore, the goal of the experiment was to demonstrate that an adversary can observe these induced traffic fluctuations, provided the anonymizing service does not degrade client–server traffic throughput.

While the download was in progress, the server increased the transmission speed gradually by 50 Mbit/s, 100 Mbit/s, 200 Mbit/s, 300 Mbit/s and 500 Mbit/s, for every iteration of the experiment. The probing nodes (`probers`) measured the available bandwidth variation on both branches. The available link bandwidth fell steadily on all routers along the path carrying `client1`'s traffic. The probing of `router5` and `client2`, along the idle network branch, resulted in very high true negatives. For brevity, we only present the results obtained by probing `router3` and `router5`. The graphs representing these are presented in Figures 3.3 and 3.4.

The available bandwidth drops as the server increases its rate to the victim, and thus occupies greater share of the available bandwidth along the path via `router3`. Probing `router5` along the path connecting `client2` to `server` shows no significant fluctuation in available bandwidth. Overall, our experiments indicate that we can indeed detect

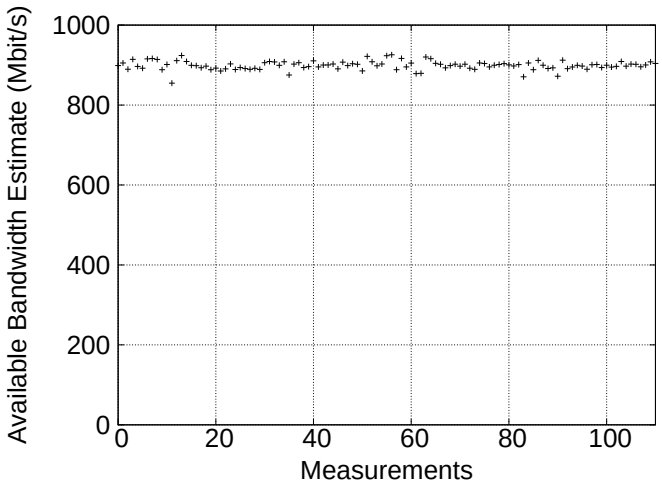


Figure 3.4: We correctly measured the absence of consistent available bandwidth fluctuation on `router5` (not in the victim's path).

small fluctuations by utilizing about 5–10% of the link bandwidth. Although this is a large value (approximately 50–100 Mbit/s), it is only a small fraction of the link bandwidth. This encouraged us to believe, that our technique will work well even in situations where only a small portion of the network link is being utilized by anonymous traffic.

We used a large file for our experiments. But we could have achieved the same fluctuation through multiple downloads of many small files. Through co-ordinated probing of the candidate links, momentary burst (due to small files being downloaded) can be easily detected. The sudden fall in available link bandwidth from approximately 100% (900 Mbit/s) to 90% (800 Mbit/s) within a short interval (few seconds) and and sudden rise later to 100%, in tandem to the induced traffic fluctuations, proves this. Further evidence of LinkWidth's effectiveness to detect small bandwidth fluctuations is presented in in our technical report [Chakravarty *et al.*, 2008c].

3.4.2 In-Lab Experiments

To further support the DETER results, we performed the same experiments in an in-lab environment using commodity machines connected via a Gigabit switched network. Figure 3.5

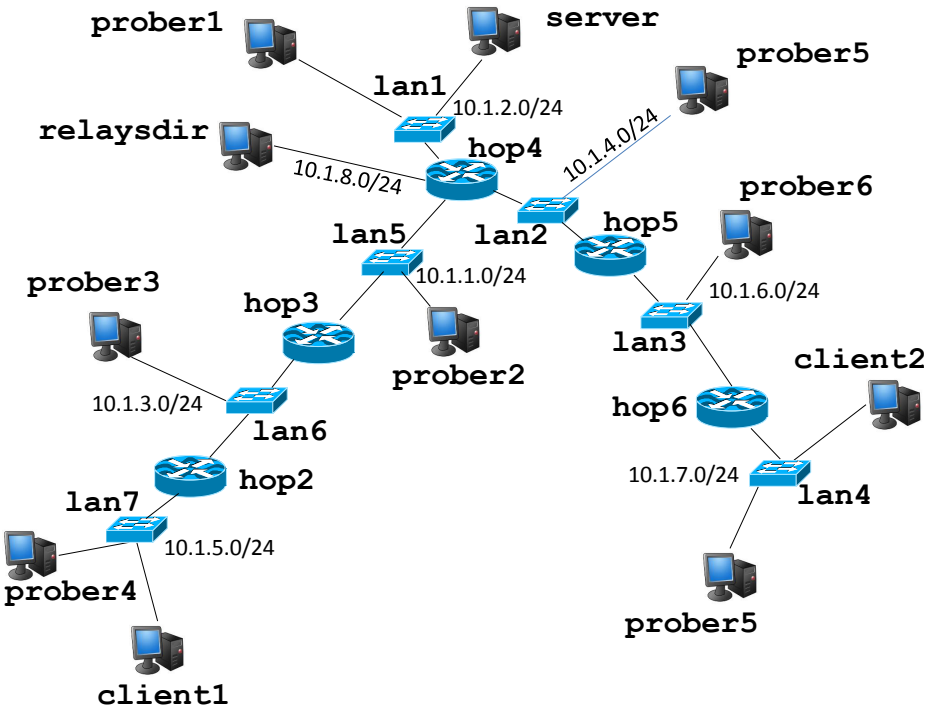


Figure 3.5: In-Lab testbed network topology.

depicts the in-lab network testbed topology used to demonstrate our technique. Again, the client `client1` is the victim and is connected to the `server` via a `squid` proxy installed on the host `relaysdirs`.

As before, the client downloaded a large file and the server varied the TCP throughput. The probing hosts measured the available bandwidth on the routers along both branches of the network - one leading to `client1`, which downloads the file, and the other leading to `client2`, which is not involved in the transfer.

Available bandwidth on all the routers along the path connecting `client1` to `server` fluctuated, as `server` varied the available TCP traffic to port 80, that it saw originating from the host `relaysdirs`. For brevity, we present graphically the results from probing `hop3` (Figure 3.6), along the path leading to the victim and `hop5` (Figure 3.7), leading to the idle node.

We observed reduction in available bandwidth as client-server traffic eventually occupies more bandwidth along the path via `hop3`. The available link bandwidth of `hop5`

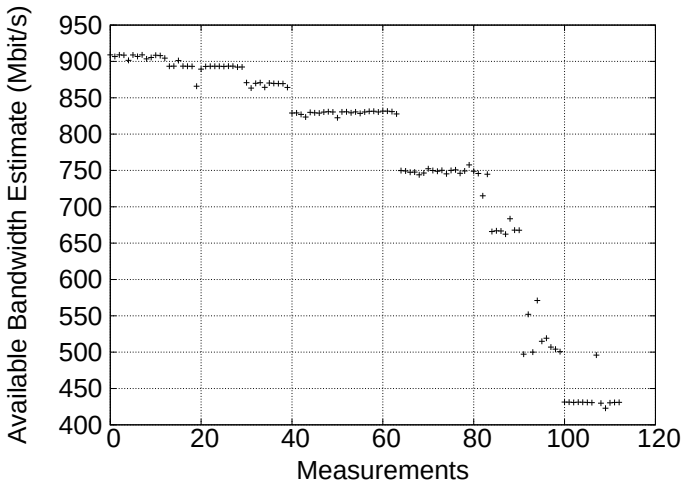


Figure 3.6: Available bandwidth on `hop3` drops uniformly when we increase the traffic towards the victim.

does not show any drastic change, as it was not along the actual download path. Probing router `hop3` and `client1` also showed similar bandwidth fluctuations while `hop6` and `client2` showed no definite fall in the available the link bandwidth; thus concurring with our idea of tracking link bandwidth fluctuation along the path leading up to the actual source of traffic.

3.4.3 Probing Tor Relays, Clients and Hidden Services

The validation of our technique in previous subsections, albeit on a controlled environment, encouraged us to believe that our technique might potentially be used to track induced available bandwidth on network routers connecting a Tor client to a Tor Entry Node. Therefore, we attempted to identify the Tor Onion Routers (ORs) participating in a real-world circuit. The server colluded with the adversary to induce fluctuations in the circuit throughput. This resulted in available bandwidth fluctuation on ORs and network routers connecting these ORs to Onion Proxies (OPs)[4]. *This experiment is elaborated in our paper* [Chakravarty *et*

[4]Onion Proxy is the term used for Tor clients [Dingledine *et al.*, 2004]

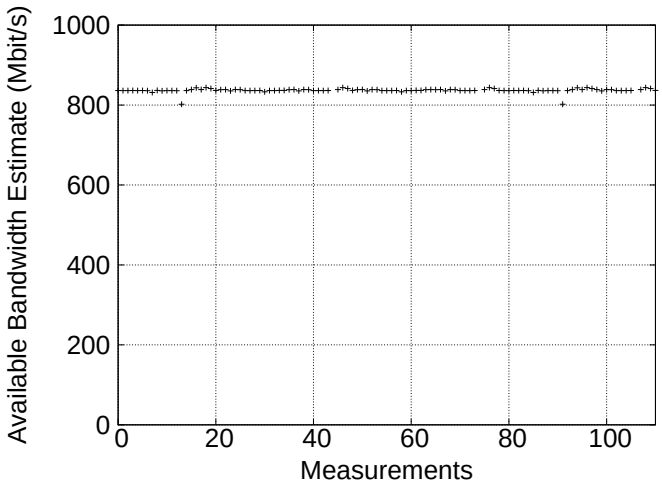


Figure 3.7: There is no persistent available bandwidth fluctuation on `hop5` (unlike `hop3`, that is along path of the download traffic).

al., 2008b]. We concisely describe the experiment and results here. Figure 3.8 illustrates how the adversary probed the Tor relays involved in a circuit.

The colluding web server transmitted a file which the client downloaded the file through a Tor circuit. During the download, the adversary used our traffic analysis technique to identify the victim relays participating in the circuit. While the server shaped the circuit's throughput to various values, the adversary measured the available bandwidth using LinkWidth. This process was repeated several times. In every iteration, the adversary changed the client's bandwidth share, increasing it each time by approximately 100 Kbit/s. The adversary detected decrease in measured available bandwidth that was reflected through increase in congestion and packet losses.

In our experiments, we successfully identified **all** the relay nodes in 11 out of the 50 circuits that we probed. For 14 circuits, we were able to identify only two of the three participating Tor relays. There were 12 circuits in which only one of the relays was detected. There were also 13 circuits that we are unable to detect any of the participating ORs. Finally, among all the 150 ORs probed there were 22 which filtered all probe traffic.

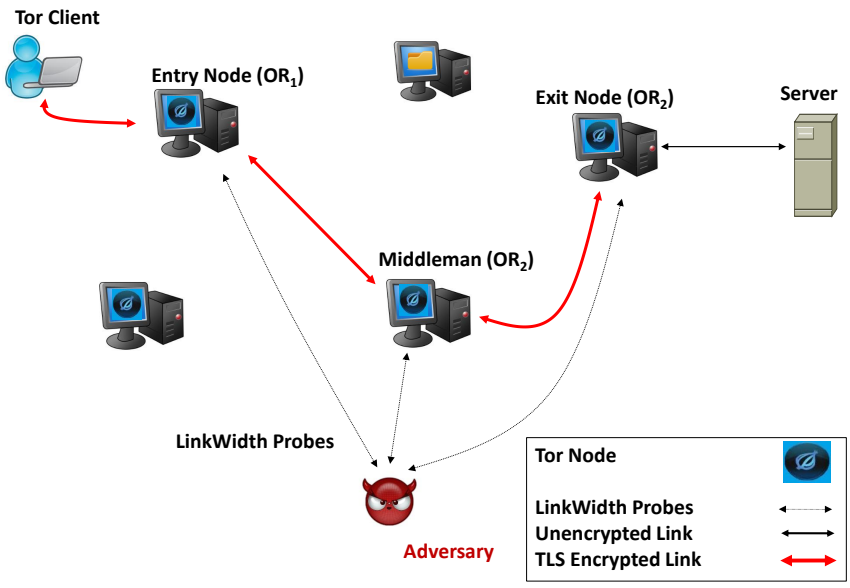


Figure 3.8: Adversary probing the fluctuations in available bandwidth of ORs participating in a Tor circuit.

A similar experiment was performed for obtaining an estimate of the false positives. Initial observations yielded approximately 10% false positives. *However, on repeated iterations of the same experiment, we detected no false positives.* These very likely result from noises and errors in our measurement techniques resulting from lack of adequate network vantage hosts, background network cross-traffic and asymmetric network links.

Probe Set-up and Technique for Identifying Tor Clients: To determine the network identities of Tor clients involved in Tor circuits, we used the same set-up as in Figure 3.8. However, in this experiment, the adversary probed routers on the network path connecting the Tor Entry Nodes to the Tor Clients. The client fetched a relatively large file from a colluding server, which shaped the bandwidth of the connection.

Lacking a "network map" that has link-level connectivity information of the Internet, we relied on `traceroute` information to discover the hops between the client and its Entry Node. However, in practice, an adversary equipped with AS-to-AS and link-level network connectivity maps, needs to probe only a relatively small set of links to determine

Circuit Number	Hops from Client –Entry Node Node	Correctly Detected Client–Entry Node Hops	Unresponsive Routers	Routers Not Reporting Enough Fluctuations	Success Rate
1	10	6	4	0	60.00%
2	15	4	0	0	26.67%
3	18	4	7	7	22.23%
4	18	5	8	5	27.78%
5	14	6	2	6	42.86%
6	14	9	1	4	64.30%
7	15	7	2	6	46.67%
8	14	7	2	5	50.00%
9	14	4	2	8	28.57%
10	15	6	4	5	40.00%

Table 3.1: Available-Bandwidth Fluctuation Detection in Links Connecting a Tor Client and its Entry Node.

which of them might be carrying the victim's traffic. Entry and exit, to and from an AS, is through a small number of *inter-domain* routers. The adversary can look up the AS location of the Tor relays from the inter-domain routing information. This will enable him to track the induced available bandwidth variation **only** on inter-domains routers and search for the AS that hosts the victim anonymous client. Nodes in all the probable ASes, with link speeds comparable to that of the inter-domain routers', could be used for this task. To obtain higher fine-grained network position, the attacker might have to track down to the end hosts. This step will require ISP router maps through services such as as *Rocketfuel* [Rocketfuel,]. *Though seemingly an exponential search problem, this would be reasonably tractable provided the fluctuations could be detected with high confidence*. Moreover, optimizing the adversary's search of links to probe is a different problem and we do not consider that aspect of the attack in our research.

The results from probing the available link bandwidth variations on network routers connecting the clients to their respective Entry Nodes, is presented in Table 3.1. The accu-

rate detection of bandwidth fluctuation was performed through the detection of packet loss changes. This loss is indication of decrease in available bandwidth, whenever the routers and Tor relays were probed using LinkWidth. Our technique used an un-cooperative method of available bandwidth or throughput estimation. Sometimes the probe traffic, being aggressive, prevented a Tor client, using TCP (which is "elastic" and non-aggressive), to utilize all of the allowed bandwidth. This lead to an "on-off" pattern in the the client's download. This is particularly true when the probes and the probed traffic traverse the same victim router.

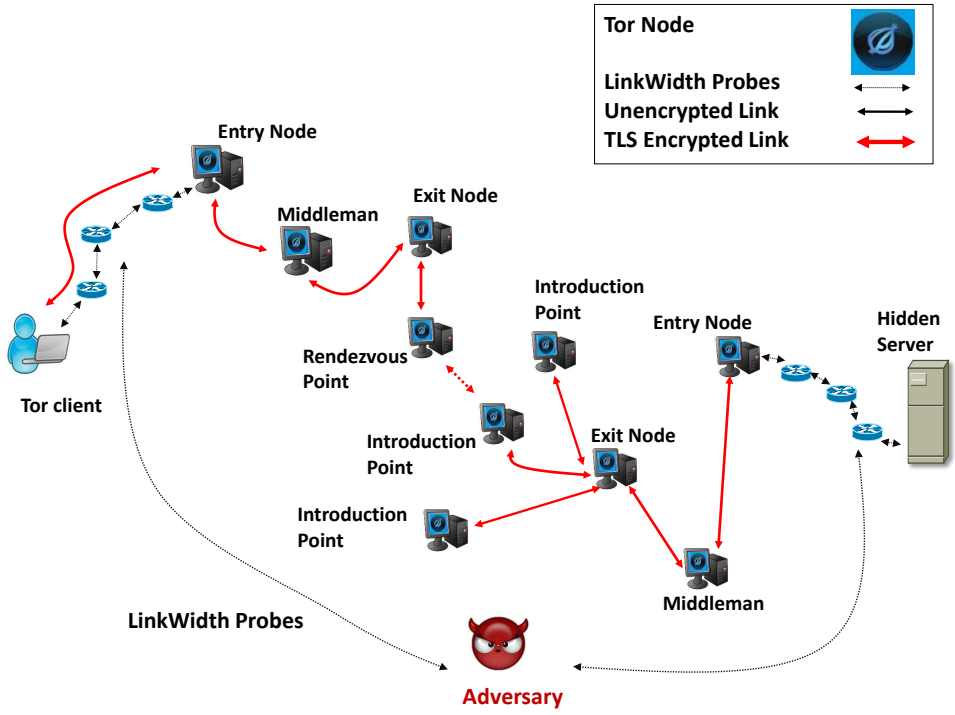


Figure 3.9: The adversary measures the available bandwidth and thus detects any fluctuations in each link of the path leading to the victim.

As a caveat, modifying the available throughput of the TCP connection through a certain repeating pattern (*e.g.,* 50 Kbit/s, 100 Kbit/s, 500 Kbit/s, 50 Kbit/s, 100 Kbit/s, 500 Kbit/s), would be akin to inducing a distinct "signal" or "watermark" in the client-server traffic. If very high true positives are assured, then this "signal" could be detected always and only on

relays and routers through which the victim's traffic travels[5]. This can optimize the search for links to probe while determining the source of the anonymous traffic[6].

Probe Set-up and Technique for Identifying Tor Hidden Servers: To identify a Hidden Service, we used the set-up depicted in Figure 3.9. Here the adversary probed the routers connecting Hidden Service to its Entry Node. Contrary to the previous cases, the available bandwidth fluctuation was induced by the client which is assumed to collude with the adversary. We relied solely on `traceroute` for determining which routers connect a Hidden Server to its corresponding Entry Node. Table 3.2 summarizes the results of this experiment:

Circuit Number	Hops from Hidden Server–Entry Node	Correctly Detected Hidden Server–Entry Node Hops	Unresponsive Routers	Routers Not Reporting Enough Fluctuations	Success Rate
1	13	4	2	7	30.70%
2	12	9	0	3	75.00%
3	11	7	1	3	63.64%
4	14	5	4	5	35.71%
5	12	9	0	3	75.00%
6	13	3	3	7	23.08%
7	16	5	4	7	31.25%
8	13	3	2	8	23.08%
9	17	4	1	12	23.53%
10	13	5	1	7	38.46%

Table 3.2: Available-Bandwidth Fluctuation Detection in Links Connecting a Hidden Server to Its Entry Nodes

In almost every circuit, there were some routers which filtered our probe packets. The

[5]Thereby also eliminating false positives and false negatives.

[6]Without such an optimization, the adversary might end up performing a depth-first search of large segments of the Internet rooted at the Tor entry node

rest of the routers were either detected correctly or not detected at all (*i.e.*, no false positives). This can be attributed to the lack of sufficient vantage points or to insufficient throughput achieved by the client in some cases (approximately 5–10 KBytes/sec). Despite of these practical problems, we were still able to trace the bandwidth fluctuations along the path (and hence the identity) of the Tor client and Hidden Server with high accuracy; over 60% and 75% in some of the circuits. The observed degradation in the client's performance whenever the adversary probed the candidate routers, are accepted as "available bandwidth fluctuations". Placing a large number of probing hosts at network vantage points would provide the adversary with better detection resolution and accuracy.

3.5 Discussions, Issues and Limitations

We initially tested our trace-back technique under various controlled environments. Our results indicate high true positives and almost zero false negatives. Small bandwidth variations, due to the introduction of a 50–100 Mbit/s TCP stream, were clearly discernible on a 1 Gbit/s link. This led us to believe that small bandwidth fluctuations on *high-capacity links* can be detected provided there is low background cross traffic that may introduce false positives or false negatives in the measurements.

LinkWidth provides very accurate available link bandwidth estimation. As presented in our technical report [Chakravarty *et al.*, 2008c], LinkWidth can accurately detect 1 Kbit/s fluctuation in available link bandwidth, in an environment free of network congestion and external disturbances. Ofcourse, this accuracy decreases when the variations decrease as a percentage of the overall link capacity. Small distortions, for instance 50 Kbit/s, on a 500 Kbit/s are easier to detect, than when they are on a 1 Gbit/s link.

Simple fluctuations on network links of the in-lab testbed could be detected within 15–20 seconds. The probing speeds were adjustable run-time parameters. Faster probing caused greater contention to the client-server traffic, thereby slightly decreasing the detection accuracy and granularity.

Having obtained high true positives under controlled environments, it seemed intuitive that an adversary could potentially detect available bandwidth fluctuation on an anonymiz-

ing proxy and its propagation to corresponding clients or servers via network routers. It is important that adversarial nodes are located at network vantage points where they can filter out traffic that causes unwanted distortion to the probes. It is also essential that a Tor client achieves high end-to-end throughput through Tor relays which is comparable to the installed link capacity of the network routers.

When applied to detect available link bandwidth variations on real Tor relays, we were able to detect with some success, fluctuations on network routers connecting the client to its respective ORs. However, we restricted our selection of Tor relays within the US, to position our US-based probing host at a better network vantage point, when probing Tor relays. Probing nodes residing across trans-oceanic links seems infeasible and provided erratic results. Consequently, we were limited by the number of Exit Nodes within the US. Out of the approximately 150 exit relays at the time of our experiments less than 100 allow clients to set-up their own circuits. Moreover, less than a fifth of these allow Hidden Servers to communicate with anonymous clients. This is mostly due to intermittent quality of service, node availability, and exit policies that restricted connectivity to a small set of permitted users. Probing Tor relays and network routers required considerably more measurements than the in-lab measurements (approximately 2–5 minutes per relay or router). High Internet cross-traffic and low Tor traffic necessitates longer probing and more measurements to avoid false positives and false negatives as much as possible.

In real world scenarios, there maybe various ways to entice a Tor client to connect to such malicious servers. Tempting commercials on a website, luring a Tor client to click on, could be one such tactic. This could download applications, like multiple Adobe Flash animations, on the client's host, resulting in a sudden change in his/her available network link bandwidth. An adversary running multiple co-ordinated probing hosts, probing suspected links, could detect such a sudden sharp change in available link bandwidths on all links connecting the anonymous party to its anonymizing service; thereby revealing its identity. The adversary would require to own only a frame of a popular website, say a blog or an online forum, visited frequently by users who wish to stay anonymous.

Apart from the lack of accuracy in detecting small variations of available link bandwidth, Reardon and Goldberg have described why current Tor circuits offer low end-to-end

throughput [Reardon and Goldberg, 2009]. This is primarily because of multiplexing many TCP streams through a single Tor circuit connecting a Tor client to a relay or between the relays themselves, if such circuits already exist. Therefore, TCP congestion control and flow control mechanics affect the performance of all Tor circuits that are multiplexed over a single TCP connection. Packet losses and reordering affecting the cells of one Tor circuit reduced the overall performance of the TCP connection. Such losses cause the cells from unrelated Tor circuits to be delayed as well.

These inherent measurement limitations can be potentially leveraged to create countermeasures or even narrow the applicability of our attack. For instance, an anonymous client can utilize parallel connections in a round-robin fashion to access the same server. This would diffuse the ability of the server to generate detectable traffic variations: traffic spikes would be distributed across all the parallel connections. Likewise, traffic smoothing by anonymizing proxies is another potential countermeasure. Tor allows relay operators to use such techniques. Another option is to use shorter circuit lifetime. This would impose some time limitations on the duration of the communication path, making it harder for an adversary to completely trace the target through the anonymizing network. Anonymous connections using longer paths by employing more relays do not appear to make the attack appreciably more difficult. However, as discussed in [Chakravarty *et al.*, 2008a], it can significantly affect the client's perception of the connection throughput.

3.6 Conclusions

In this chapter, we proposed a new traffic analysis technique that can used to confirm the network identity of anonymization relays and end-points (users) of low-latency network anonymity systems. Our technique involves an adversary who can probe links from many network vantage points using single-end controlled bandwidth estimation tools. In addition, the adversary employs a colluding server or is able to otherwise induce fluctuations in the victim's TCP connection. It is not essential to own such colluding servers: using carefully crafted online ads and pop-ups can be used judiciously to trick users to click on specific links and result in traffic spikes. Using the vantage points, the adversary measures the

effects of this fluctuation as it "trickles" through the anonymizing relays and intermediate routers to the victim.

Our approach worked well in an controlled environments, free of external network and system disturbance and congestion, and where the end-to-end throughput of the anonymizing network allows for bandwidth variations that can be detected by the vantage points. This motivated us to test our attack technique in real-world Tor circuits. Our experiments show that we were able to expose real-world Tor relays with a true positive rate of 59.46%. Furthermore, we could successfully traceback to the Tor clients and Hidden Servers by probing the network routers connecting them to their Entry Nodes. On average, we could correlate 49.5% of the network routers to the victim Tor traffic that they carried. Further correlations were not always feasible due to bandwidth limitations imposed by relay operators and Tor's poor performance owing to circuit scheduling and management [Reardon and Goldberg, 2009]. We believe that our work exposes a real weakness in proxy-based anonymity schemes. *This threat will become increasingly more apparent and accurate as future networks and hosts participate in higher end-to-end throughput circuits.* In the next chapter we study a traffic analysis attack involving data obtained from traffic monitoring systems installed in existing network infrastructure (e.g. Cisco's NetFlow).

Chapter 4

Traffic Analysis Against Anonymity Networks Using Flow Records

4.1 Overview

In the previous chapter we focused on our traffic analysis attack to confirm the identify the victim within the anonymity set, using our single-end controlled bandwidth estimation tool. In this chapter we explore yet another traffic analysis attack that involves evaluating the effectiveness of using data from existing network monitoring infrastructure, installed in commodity networking hardware, such as Cisco's NetFlow, to launch a traffic analysis attack for identifying the source of anonymous traffic.

We described earlier that de-anonymization of an anonymous client is a two step process. The first step involves finding the anonymity set, and the second step involves finding the victim within this anonymity set. In the past, various research efforts have explored ways to find the anonymity set that includes the anonymous victim. Some have suggested advertising fake bandwidth to attract traffic [Bauer *et al.*, 2007; Øverlier and Syverson, 2006]. Others have tried to study the distribution of autonomous systems to find those that probably intercept random paths going to both entry and exit nodes from various client and server locations [Feamster and Dingledine, 2004; Edman and Syverson, 2009a]. They showed that even a single AS may observe about 22% of randomly generated Tor circuits. Others such as Zieliński et al. [Murdoch and Zieliński, 2007] (and more recently Johnson et

al. [Johnson *et al.*, 2013], have tried to demonstrated that a small set of Internet Exchanges (IXes) could monitor large number paths leading to and from entry and exit nodes.

Packet-level traffic monitoring at such a scale would require the installation of passive monitoring sensors capable of processing tens or hundreds of Gbit/s traffic. Although not impossible, setting up a passive monitoring infrastructure of this scale is a challenging endeavor in terms of cost, logistics, and effort. An alternative, more attractive option for adversaries would be to use the readily available (albeit less accurate) traffic monitoring functionality built into the routers of major Internet Exchanges (IXs) and Autonomous Systems (ASs), such as Cisco's NetFlow, as proposed by Murdoch and Zieliński.

NetFlow records traffic statistics corresponding to abstractions called *flows*. A flow corresponds to a TCP or UDP socket, consisting of source and destination IP addresses and port numbers. Compared to packet level traffic monitoring, the flow level aggregation of the measured traffic properties and the use of aggressive sampling make NetFlow data less than ideal for traffic analysis attacks. Zieliński et al. showed, through simulation, that traffic analysis using sampled NetFlow data is possible, provided there are adequate samples. However, their efforts did not explore the practical feasibility and effectiveness of using a sub-system like NetFlow to determine the source of anonymous traffic. *Is using NetFlow practical? Can an adversary accurately de-anonymize a Tor client solely using NetFlow data?* These are some of the question which we try to answer through our research.

As a step towards answering these questions, we present a practical active traffic analysis attack against Tor, that relies on NetFlow statistics. Our approach is based on identifying pattern similarities in the traffic flows entering and leaving the Tor network using statistical correlation. To alleviate the uncertainty due to the coarse-grained nature of NetFlow data, our attack relies on a server under the control of the adversary that introduces deliberate perturbations to the traffic of anonymous visitors. Such an adversary is similar to the one described in the previous chapter. Among all entry-node-to-client flows, the actual victim flow can be distinguished due to its high correlation with the respective server-to-exit-node, as both carry the induced traffic perturbation pattern.

We evaluated the effectiveness of our traffic analysis attack first in an in-lab environment, and later using a set-up involving real Tor network relays. For our research, we used

Start	End	Sif	SrcIPaddress	SrcP	DIf	DstIPaddress	DstP	P	Pkts	Octets
23:59:06	23:59:36	65535	192.168.0.20	50000	2	213.163.65.50	54089	6	3	156
23:59:06	23:59:36	2	213.163.65.50	54089	65535	192.168.0.20	50000	6	3	1914
23:59:29	23:59:30	2	71.58.107.145	42259	65535	192.168.0.20	50000	6	10	3961
23:59:29	23:59:30	65535	192.168.0.20	50000	2	71.58.107.145	42259	6	9	3578
23:59:33	23:59:33	65535	127.0.0.1	55171	1	127.0.0.1	10002	17	1	1492

Table 4.1: Sample NetFlow Records Showing Various Fields

a combination of flow data gathered from open source tools such as `ipt_netflow` [NetFlow iptables module,] and `flowtools` [`flow-tools`,], as well as the flow records from our institutional Cisco router. In our controlled in-lab experiments, we relied solely on data from open source tools, while in the experiments involving our public Tor relay, we used data both from open source tools as well as from our institutional edge router. In the controlled in-lab environment we had 100% success in determining the source of anonymous flows. When evaluating our attack with traffic going through the public Tor relay, we were able to detect the source in 81.6% cases. We observed about 12.7% false negatives and 5.5% false positives in our measurements.

We begin this chapter with a brief description of how NetFlow traffic motoring works. This is followed by a description of the threat model and our attack strategy. Thereafter we describe how we experimentally evaluated our attack and finally conclude the chapter with a brief description of the accuracy and limitations of our approach.

4.2 Network Monitoring (NetFlow)

NetFlow is a network protocol designed for collecting and monitoring network traffic. NetFlow groups exchanged data into *flows*, which correspond to TCP connections or other IP packets sharing common characteristics, such UDP packets sharing source and destination IP addresses, port numbers, and other information (see protocol specification [Claise,] for further details).

Table 4.1 presents some sample flow records. The columns labeled `Start` and `End` denote flow start and end times. `Sif` and `Dif` represent the SNMP source and destination

interface `ifIndex` respectively. `SrcIPaddress` and `DstIPaddress` denote source and destination IP addresses, and, similarly, `SrcP` and `DstP` denote port numbers. `P` is the protocol (6 represents TCP and 17 represents UDP). `Pkts` and `Octets` represent the number of packets, and bytes respectively, observed for this flow in the particular time frame.

Most Cisco routers these days include networking monitor capabilities, such as NetFlow. Other major router and networking device manufacturers have their own, similar protocols, like Juniper `jflow` [`J-Flow`,], Huawei `Netstream` [`Netstream`,], and Alcatel Lucent `sflow` [`sFlow`,]. There are also various open source implementations for collecting network statistics using NetFlow, such as `ipt_netflow` [NetFlow iptables module,], `fprobe` [`Fprobe, `] and `flow-tools` [`flow-tools, `].

The way these systems work is by creating a NetFlow record in memory and updating it as more packets are forwarded through the router. Each flow is associated with two configurable timers, namely the *active* and *inactive* timers. If data has been recorded, for a flow, within the recent active timeout period, signalled by the expiration of the `'active'` timer, then it classified as an `'active'` flow and aged out to persistent storage. Thereafter, fresh records and entries are instantiated in the flow cache. Complementarily, a flow that hasn't recorded activity within the recent inactive timeout period, is labelled as an `'inactive'` flow. Following the expiration of the inactive timer, the flow record is moved to persistent storage and new records, corresponding to the flow, are instantiated. These timers decide the length of the flow intervals. For example, if the `'active'` timeout is 10 seconds, and if a flow corresponds to a file download session, and records more or less continuous activity, then the flow records have time intervals that are 10 seconds long. The records are flushed-out to persistent storage, so they can be processed to produce higher-level statistics, generate traffic reports, and so on, based on their age, or when a flow is terminated. For instance, when a TCP FIN packet has been observed for a TCP connection. Administrators can modify various system parameters to customize the process [Flexible NetFlow Command Reference,], like setting timeouts for controlling when to move active and inactive flow records to persistent storage, and enabling traffic sampling. As Internet speeds have grown, vendors introduced packet sampling to minimize the overhead of network monitor-

ing in packet forwarding, in exchange for accuracy in the reported flow records.

4.3 Approach

4.3.1 Threat Model

In our research, we mainly focus on the problem of evaluating the effectiveness of using NetFlow data to perform practical traffic analysis attacks to identify the source of anonymous communication. In our attack model, we assume that the victim is lured to access a particular server through Tor, while the adversary collects NetFlow data corresponding to the traffic between the exit node and the server, as well as between Tor clients and the victim's entry node. The adversary has control of the particular server (and potentially many others, which victims may visit), and thus knows which exit node the victim traffic originates from.

We assume a powerful-enough adversary that can monitor traffic at various network locations, allowing the inspection of Tor traffic towards a significant number of entry nodes [Edman and Syverson, 2009a; Murdoch and Zieliński, 2007; Feamster and Dingledine, 2004]. The adversary could identify important ASes and their topological relationships using methods similar to those presented by Schuchard et al. [Schuchard *et al.*, 2012]. Alternatively, the adversary could use methods similar to those presented in [Evans *et al.*, 2009; Mittal *et al.*, 2011; Chakravarty *et al.*, 2008b] to identify the entry node of a particular victim circuit. The challenge for the adversary is to determine the real identity of the anonymous client that corresponds to a connection seen at the server, using solely NetFlow data from the vantage points of i) the exit node towards the server, and ii) the various clients towards entry nodes. *Having determined the identity of the entry node involved in the victim anonymous connection, the adversary needs to find the entry-to-client flow that uses this entry node and which correlates closely to the server-to-exit flow.*

4.3.2 Attack Approach

In our attack, a victim is lured to fetch some data from a server under the control of the attacker, while the server perturbs the traffic of the TCP connection originating from an

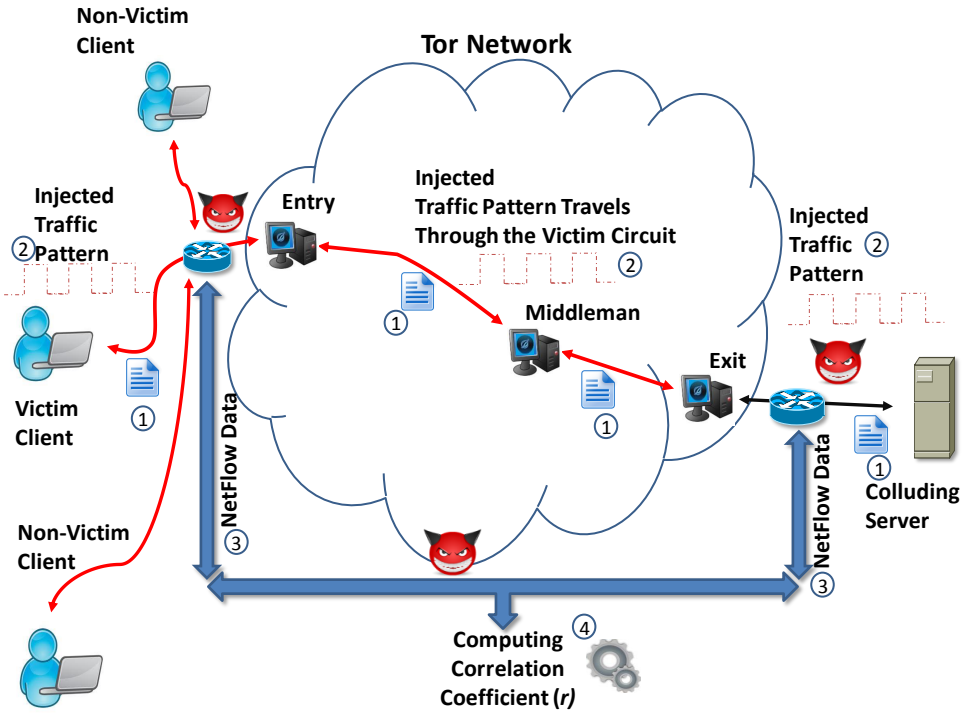


Figure 4.1: Overall Process for NetFlow Based Traffic Analysis Against Tor The client downloads a file from the server ①, while the server injects a traffic pattern into the TCP connection it sees arising from the exit node ②. After a while, the connection is terminated and the adversary obtains flow data corresponding to the server to exit and entry node to client traffic ③, and computes the correlation coefficient between the server to exit traffic and entry to client statistics ④.

exit-node. The server "injects" a specific traffic pattern that aids the identification of the victim flow among several other possible ones. As shown in Figure 4.1, after the transfer ends, the adversary obtains the flow records of all the client-to-entry-node connections that were monitored during the same time period (at one or more entry nodes), and computes the correlation coefficient between these flows and the given exit-node-to-server flow. Correlation is performed using solely the reported transferred bytes, which is the only relevant traffic statistic available from flow records.

Various factors, such as the values of the *active* and *inactive* flow cache eviction timers, and the inherently bursty nature of web traffic commonly result in an inadequate number

of flow samples that what is ideally required for computing the correlation coefficient. The longer the duration of the fingerprinted transfer, the higher the chances that enough flow samples will be gathered. In our experiments we assume that the victim downloads a large file (in the order of tens of megabytes) that generates sustained traffic for a relatively long duration, between 5–7 minutes. Depending on the capabilities of the involved routers, however, the same accuracy could be achieved using shorter data transfers.

A victim can be enforced to download a file from a particular server in many ways. An adversary could for instance inject (invisible) IFRAMEs in web sites frequented by the targeted victims, target particular victims using social engineering tactics (e.g. "spear-phishing" attack), or advertise decoy multimedia files, e.g., pirated movies.

Implementation In our prototype implementation, the server fluctuates a client's traffic using Linux Traffic Controller [Hubert *et al.*,], and we have explored two different kinds of traffic patterns. The first is a simple "square wave" like pattern, achieved by repeatedly fluctuating the victim's transfer rate between two values. A second, more complex "step" like pattern is achieved by repeatedly switching between several pre-determined bandwidth values in an arbitrary order. These different perturbations help evaluate our attack accuracy through both simple and complex injected traffic patterns.

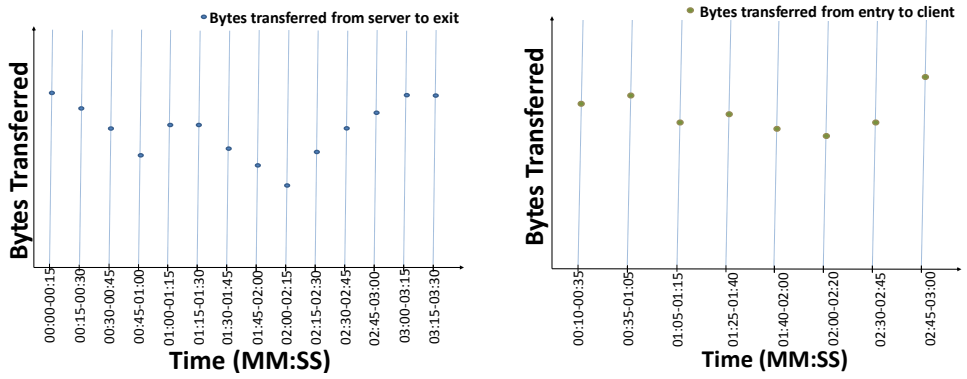
For our initial in-lab experiments, flow records were generated and captured using the open source tools `ipt_netflow` [NetFlow iptables module,] and `flow-tools` [`flow-tools`,], respectively. In such a controlled environment, free of network congestion and external system disturbances, our approach achieved 100% success in determining the source of anonymous connections. Thereafter, we moved to experiments to test the effectiveness of our attack against real Tor traffic. For these tests, we obtained data from a public Tor relay serving hundreds of Tor clients. The flow records for the server-to-exit traffic were generated and captured using the aforementioned flow tools. The flow records for the entry-to-client traffic were generated first using the flow tools running on the same host as the entry node, and later by our institutional edge router. In the latter case, the flow data from the router was often sparse, due to the use of aggressive sampling, and multiple intervals were typically aggregated into a single flow record. This generally happens due

to the combination of flow expiration timeout values and the router's network load. Such aggregation is not deterministic and it is difficult to divide a large interval into smaller ones without the knowledge of the ordinate values of the aggregated intervals. We thus devised the following strategy to rectify such flows and correctly align them to exit-to-server flows.

To correctly align the flows, we first compute the average traffic throughput in each interval of the server-to-exit and entry-to-client flows. Thus, if $\texttt{Bytes}_1$ bytes are transferred in an interval (t_1, t_2), then the average throughput of that interval is computed as $(\frac{\texttt{Bytes}_1}{t_2 - t_1})$. Each time interval of the server-to-exit flows is then divided into steps of one second. For example, an interval (t_1, t_2), after the division, would result in points t_1, $t_1 + 1$, . . ., $t_2 - 1$, t_2. The throughput at each of these points is equal to the average throughput, computed as above. Thereafter, we align the start and end times of each of the entry-to-client flow intervals with the times obtained by dividing the server-to-exit flow intervals, and ignore the time values that do not align. Finally, we compute the correlation for the throughput, in the two time-aligned sets. This gives us an approximate correlation for the server-to-exit and entry-to-client traffic, even with sparse or misaligned data. Figures 4.2(a) and 4.2(b), schematically present sample flow data for server-to-exit traffic and entry-to-client traffic obtained from open-source tools and our edge router, respectively. The figures show more data points for the server-to-exit traffic data, obtained from open-source packages and sparse information for the entry-to-client traffic obtained from the flow records obtained from NetFlow data. Figure 4.3 schematically shows how the server-to-exit traffic data is aligned against entry-to-client data using our rectification strategy. The crosses represent average server-to-exit throughput values that align with entry-to-client throughput points, represented by the dots.

4.4 Experimental Evaluation

We evaluated our traffic analysis method first using and in-lab testbed and later through experiments involving data from a public Tor relay. The former experiments we conducted to evaluate the accuracy of our correlation based traffic analysis method in an in-lab set-up in the absence of Internet traffic congestion and related network and system artifacts. In the



(a) Server-to-exit traffic: Data obtained using open-source tools

(b) Entry-to-client traffic: Data obtained using Cisco NetFlow

Figure 4.2: (a)Server-to-exit traffic data obtained from open-source tools. (b)Entry-to-client traffic data obtained from NetFlow records.

experiments, our traffic analysis method detected the victim client in all cases. Thereafter, we evaluated the effectiveness of our attack through experiments that involved data obtained from a public Tor relay. These experiments involved both dense and sparse data and various effects of network congestion and path characteristics. Even under such conditions, we were able to identify the victim client in about 81.6% of the experiments (with about 5.5% false positives).

4.4.1 Experimental Evaluation in Controlled Environments (Using In-Lab Testbed)

The first in-lab experiment consisted of evaluating the effectiveness of our traffic analysis attack in an in-lab set-up with a private Tor network. The set-up used is shown in Figure 4.4.

In this set-up thirty clients simultaneously communicated to the server through circuits using the relays of the private Tor network. In reality, these clients were actually hosted on two PCs. Each PC hosted fifteen clients. The PCs were connected on the same LAN using a 10/100 Mbit/s Ethernet switch (all the client IP addresses were in the same subnet). The individual network connections of the middlemen were on different subnets (corresponding to the incoming and outgoing network connections). The outgoing network connections of

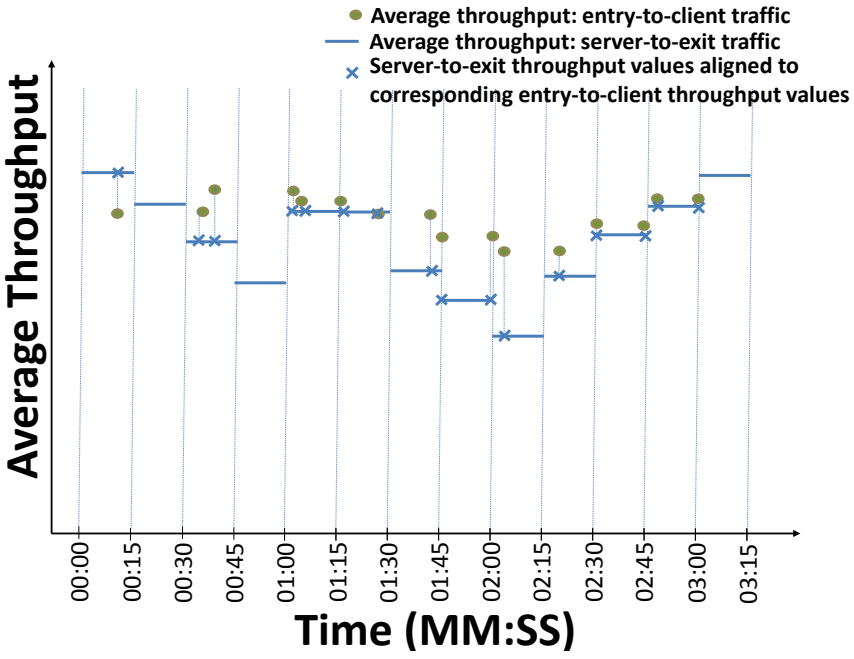


Figure 4.3: Server-to-exit and entry-to-client rectified and aligned using our rectification strategy.

the middlemen were connected to two different interfaces of the exit node which connected to the server through a separate network interface, on yet another different LAN (on yet another separate subnet). Fifteen of these clients used the circuits involving the entry node, the middleman 1 and the exit node and communicated to the server. The remaining fifteen used circuits via middleman 2 (instead of middleman 1) to communicate to the server.

We restricted the number of clients to 30. From our initial experience with this set-up, we realized, that when using Tor circuits, due to the packetization and Tor's traffic scheduling, each client obtained roughly 2.5 Mbit/s throughput. This seemed adequate for testing the effectiveness of attacks in an in-lab set-up in the absence of any kind of external network and system disturbances.

These clients downloaded large files from the server, which selected one of the clients and perturbed its traffic thereby injecting a traffic pattern. In our experiments the server injecting two kinds of patterns. The first was a "square-wave" like pattern with amplitude of 1 Mbit/s. The server achieved this by repeatedly switching the victim's traffic between

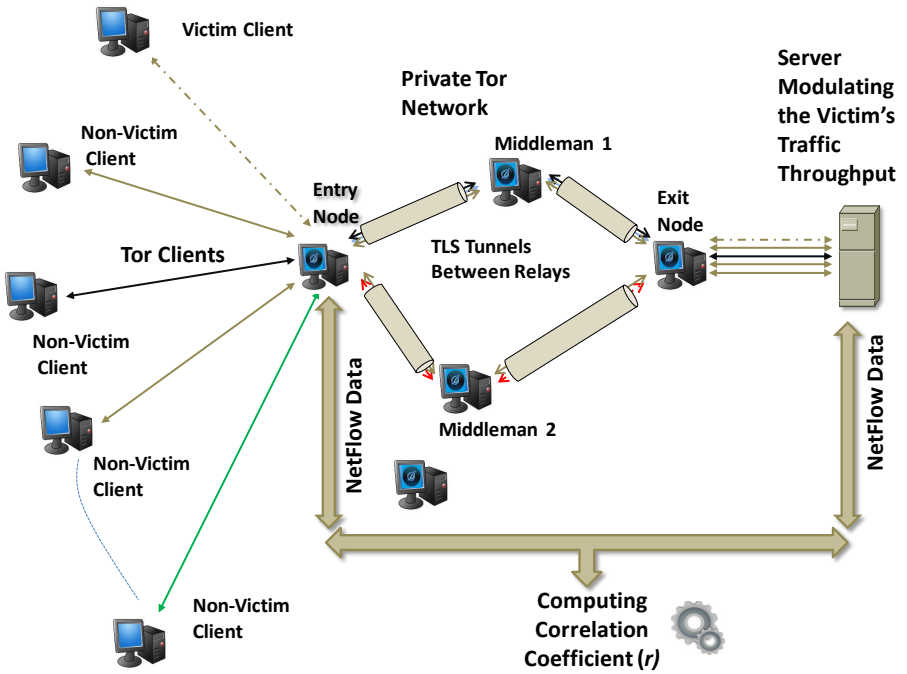


Figure 4.4: In-lab testbed used to evaluate effectiveness of NetFlow traffic analysis method.

2 Mbit/s and 1 Mbit/s, every 10 seconds. The second involved the server inject a complex "step" like pattern by switching the exit node to server TCP connection throughput through 2 Mbit/s, 1 Mbit/s, 256 Kbit/s and 512 Kbit/s, every 10 seconds. A sample plot, corresponding to the server-to-exit, entry-to-victim and non-victim clients' throughput pattern, derived from NetFlow records is shown in Figure 4.5. The complex traffic pattern is evident through a close observation of the figure. The server switches the traffic throughput through the said values. Since all the non-victims were using the same entry and exit nodes, they were affected by the same traffic schedules. Since all the circuits are involved in bulk download, they are scheduled with similar priority [Tang and Goldberg, 2010]. Their traffic throughput patterns, being closely synchronized, highlights this fact.

These experiments were repeated 60 times, 30 times corresponding to the scenario where the server injected the "square-wave" pattern and the 30 times corresponding to "step" like pattern. The experiments ran for 400 seconds. In case of "square-wave" like pattern, we observed very high correlation between the server-to-exit and entry-node-to-victim

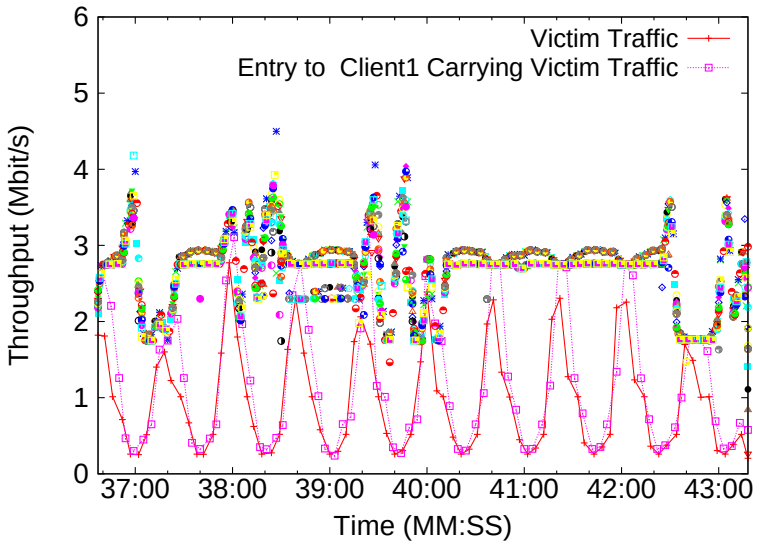


Figure 4.5: Server induced "step" like pattern exactly matches the flow going towards the victim client. The points corresponding to the flows are highlighted by connecting the sample points. The remaining points correspond to 29 other non-victim flows.

client traffic (μ:0.80 , σ:0.08) and much lower correlation, corresponding to non-victim clients (μ:0.06,σ:0.16). Similarly for the "step" like pattern, we observed a very high correlation corresponding to the server-to-exit and entry-to-victim client traffic (μ:0.92 , σ:0.06) and much lower correlation corresponding to non-victim clients (μ:0.07,σ:0.14). We observed high correlation between the server-to-exit traffic and entry-node-to-victim client traffic statistics (μ:0.90,σ:0.06), even when the experiment lasted for a shorter duration of 200 seconds.

4.4.2 Experimental Evaluation Involving Public Tor Relays

Having evaluated the accuracy of our traffic analysis attack in an in-lab environment we evaluated it using real Tor traffic, obtained from our public Tor relay that served hundreds of Tor circuits simultaneously. The set-up used is as same as the one shown in Figure 4.1. The victim clients were hosted on three different planetlab locations, namely, Texas (US),

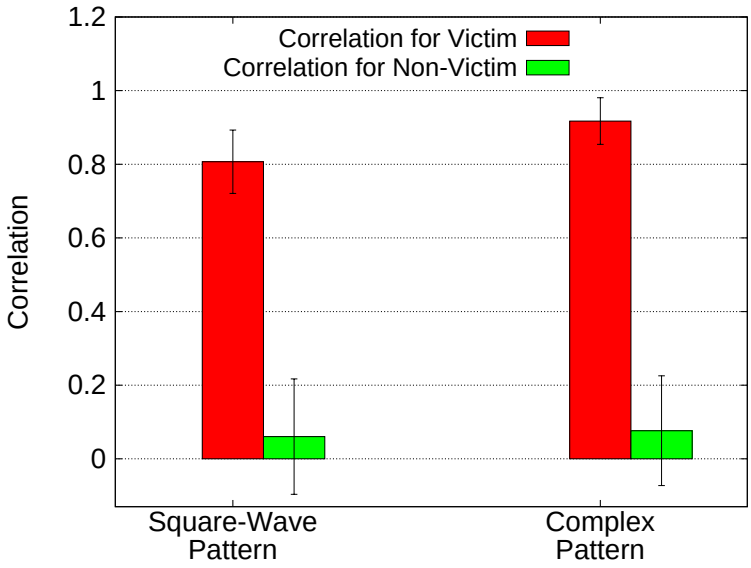


Figure 4.6: Average correlation for the server induced traffic pattern and the entry-to-victim client traffic and the maximum correlation between the server induced traffic and non-victim clients, when the server injected "square-wave" and "step" like pattern (number of clients:30).

Leuven (Belgium) and Corfu (Greece). They communicated, via Tor circuits through our relay, to a server under our control, in Spain. Our relay was chosen as the entry node. The victim clients downloaded a large file from the server that perturbed the arriving TCP connection's traffic, thereby deliberately injecting a traffic pattern in the flow between the server and the exit node. The process was terminated after a short while and we computed the correlation of the bytes transferred between the server and the recently terminated connection from the exit node and the entry node and the several clients that used it, during this interval.

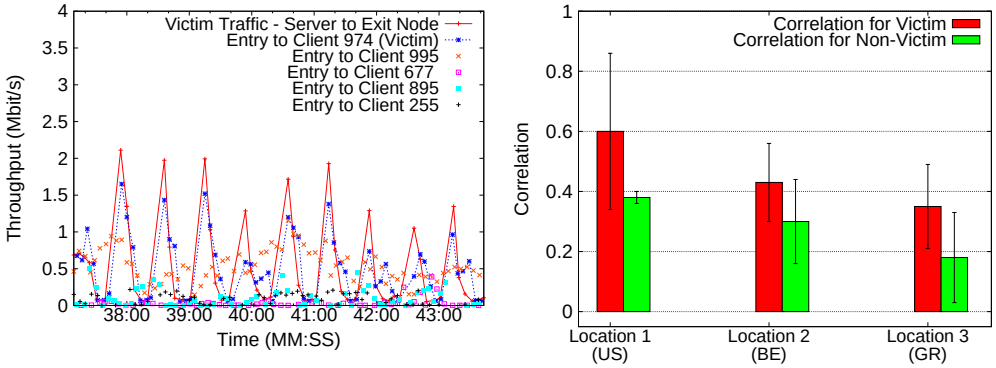
These experiments were divided into two phases. The first consisted of evaluating the effectiveness when gathering data from open-source NetFlow packages. The second part involved sparse data obtained from our institutional Cisco router.

Using Open Source NetFlow Tools

The first experiment involved the server injecting a "square-wave" like traffic pattern by freely switching the server-to-exit traffic bandwidth between approximately 2 Mbit/s and 30 Kbit/s, thereby injecting a traffic pattern with amplitude of roughly 2 Mbit/s. We used the set-up shown in Figure 4.1. The data was gathered from the server and from the entry node, using open source NetFlow packages. Each such experiment, involving the server switching between these bandwidth values, lasted for about 6 minutes and 40 seconds. Figure 4.7(a) presents sample traffic throughput variations for five flows, corresponding to one such experiment. These five flows are most correlated to the server-to-exit victim flow carrying the injected traffic pattern. The victim flow had the highest correlation coefficient of 0.83, while the one with second-highest correlation, corresponding to a non-victim client, was 0.17. A total of 1104 client were using the entry node at the time of the experiment. The figure also shows the fluctuating "square-wave" traffic pattern with an amplitude of approximately 2 Mbit/s. The server injected this pattern by switching the throughput between 2 Mbit/s and 30 Kbit/s, every 20 seconds. We computed the correlation coefficient for the bytes transferred between the server and the exit node and each of the entry node to client flows, corresponding to the duration of the experiment. Thereafter, the client that was most correlated to the server-to-exit traffic was selected as the victim.

These experiments, involving injection of "square-wave" pattern, were repeated 45 times (15 times corresponding to each of the three victim client location). Average correlation between the server-to-exit traffic and entry node-to-victim client traffic was 0.60 (σ:0.26), 0.43 (σ:0.13) and 0.35 (σ:0.14) for each of the clients at Texas (US), Leuven (Belgium) and Corfu(Greece) respectively (see Figure 4.7(b)). These corresponded to the client flows that were most correlated to the server-to-exit flow carrying the traffic pattern. The average of second-highest correlation coefficients, corresponding to non-victims, were 0.38 (σ:0.02), 0.30 (σ:0.14) and 0.18 (σ:0.15), respectively for each of the client locations.

Similar experiments were also conducted with the server injecting a "step" like pattern. To achieve this, the server switched the server-to-exit traffic between roughly 1 Mbit/s, 50 Kbit/s, 300 Kbit/s and 100 Kbit/s, every 20 seconds. This pattern was repeated several times. Figure 4.8(a) shows one such sample where the server injected this "step" like



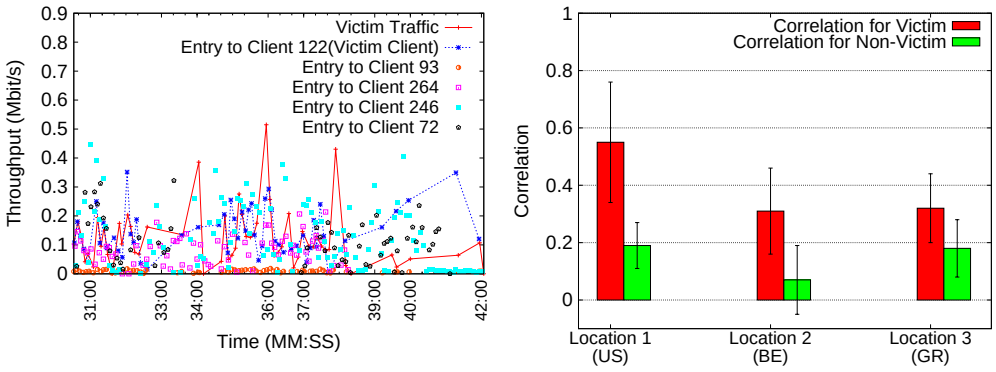
(a) Square-wave pattern

(b) Average correlation: Square-wave pattern

Figure 4.7: (a)Victim flow with a server-induced "square-wave" like pattern. The remaining points correspond to the non-victim flows with the four highest correlation coefficients.(b)Average Pearson's Correlation between server injected "square-wave" like pattern and the victim and non-victim flows for the different planetlab client locations.

pattern. In this experiment, the client whose statistics were most correlated to the server-to-exit traffic (correlation coefficient of 0.84) corresponded to the victim. The client having the next highest correlation (corresponding to a non-victim) was 0.25. A total of 874 clients were using the entry node at the time of the experiment. These experiments were repeated 45 times (15 times corresponding to each of the three client location). Average correlation between the server-to-exit traffic and entry node-to-victim client traffic was 0.55 (σ:0.21), 0.31 (σ:0.15) and 0.32 (σ:0.12), for each of the clients at Texas (US), Leuven(Belgium) and Corfu(Greece) respectively (see Figure 4.8(b)). These also corresponded to the flows having the highest correlation coefficients. The average of second-highest correlation coefficients, corresponding to non-victims, were 0.19 (σ:0.08), 0.07 (σ:0.12) and 0.18 (σ:0.10), corresponding to the victim clients at Texas (US), Leuven(Belgium) and Corfu(Greece) respectively.

In most of the experiments, we observed a clear separation between the correlation of the server-to-exit node traffic (carrying the induced traffic pattern) and the entry node-to-victim client traffic, and that measured between the former and non-victims' traffic. However, the average correlation of the injected pattern for the victim traffic was lower than what we observed in the in-lab tests. This is because the traffic pattern is distorted when it



(a) Step pattern

(b) Average correlation: Step pattern

Figure 4.8: (a)Victim flow with a server-induced "step" like pattern. The remaining points correspond to the non-victim flows with the four highest correlation coefficients.(b) Average Pearson's Correlation between server injected "step" pattern and the victim and non-victim flows for the different planetlab client locations.

leaves the Tor entry node and proceeds towards the victim client, thereby reducing the correlation of the injected traffic pattern with the entry node to victim client traffic. Further, we also found four instances where the correlation of the injected traffic with the victim traffic (from the entry node to the victim client) was lower than the correlation with some other non-victim clients' traffic. Such false negatives and false positives are primarily a combined effect of the background network congestion and routing in Tor relays, wherein the available bandwidth is distributed equally amongst all circuits [Tang and Goldberg, 2010]. Moreover, Pearson's correlation coefficient is known to be very sensitive to minor changes in input values. Small changes to input can drastically change the value of the correlation coefficient.

For each of the clients, we also computed the average bandwidth variation of the traffic for the duration of the experiment. Each of these average bandwidth variation values were subtracted from the average bandwidth variation of the server-to-exit traffic. For the victim traffic, this difference is often amongst the smallest. Thus, while correlation is sensitive to small variations in input, such a heuristic, involving the calculation of the difference between the victim traffic and server-to-exit traffic, carrying the induced pattern, can be used

to filter out flows that could lead to inaccurate correlation values arising out of lack of adequate values. In fact, in the next subsection we show how this heuristic could be used to filter out flows which can possibly result in inaccurate correlation between server-to-exit victim traffic and entry node-to-client traffic flows, when obtained from our institutional Cisco router. The records gathered from the Cisco router were sparse and resulted in inaccurate correlation coefficient computation. However, the difference between the victim-to-entry traffic and server-to-exit traffic, corresponding to the duration of the experiment, is ideally expected to be the least. We used the above observation to filter out flows which are unlikely to correspond to the victim flow. After filtering out the flows, we applied our approximation technique (described previously) to align corrected flow information and compute the correlation coefficient. We pick out the one which shows highest correlation (statistically significant, i.e. ≥ 0.2) amongst this filtered set.

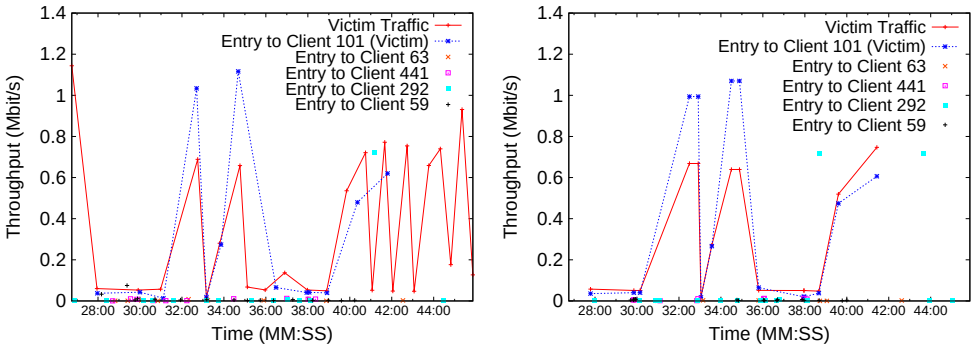
Using data from Cisco router

After evaluating our traffic analysis attack with data from open source NetFlow packages, we moved to evaluating it with data from our institutional edge router. We used the same experimental set-up as that used above to test our attack using data obtained from open source packages. The differences here was that the entry node-to-client data was gathered from the router instead of directly collecting it from the entry node. The router was configured with an active timeout of 60 seconds and inactive timeout 15 seconds. We had no authority to modify these values as the router served large fraction of our institutional network traffic (several tens of thousands of competing flows at any given point of time). We thus configured the NetFlow packages on the server with these values. As already described in the previous section, the data obtained from the router seemed much sparse and non-uniformly aligned compared to the flow records from server-to-exit node. We thus applied our approximation strategy (described in the previous section) to align the flows. The strategy primarily operates by interpolating approximate bandwidth values. The rectified flow values were then directly used as input to the correlation coefficient computation formula and the correlation coefficients between the server-to-exit traffic and entry node-to-client traffic were determined.

These experiments were essentially the same as those described in the previous subsection. The first experiment involved the server injecting a "square-wave" like traffic pattern with an amplitude of approximately 1 Mbit/s. However, unlike the experiments in the previous subsection, the server switched the throughput every 30 seconds, instead of 20 seconds. This enabled us to capture adequate samples ($\geq$ 10) for computing the correlation coefficient. This was done solely to compensate for the lack of samples obtained when the experiments ran for a shorter duration of 20 seconds (as previously). For example, in experiments involving the server injecting the "square-wave" like pattern, if the server switched the server-to-exit bandwidth every 20 seconds, and if this was repeated 10 times, then the total experiment ran for about 7 minutes, the institutional router, on an average, had only three sample intervals corresponding to the entry node to client traffic. This was primarily an effect of aggregated sampling, possibly done for sampling under heavy network loads. Traffic statistics of multiple flow intervals were accumulated into a single interval, long enough to correspond to several smaller flow intervals. Figure 4.9(a) presents a sample bandwidth variation pattern for the server-to-exit traffic and the entry node to client traffic when the server injects the "square-wave" pattern with an approximate amplitude of about 1 Mbit/s. It shows server-to-exit traffic with more data points and fewer entry-to-client points. Figure 4.9(b) presents the same data pattern after it has been rectified using our approximation strategy. As evident, here the server-to-exit traffic and entry-to-client traffic to have equal number of points and are aligned together.

We eliminate flows whose average throughput variation was not comparable to that of the server-to-exit traffic throughput variation. To do this, we computed the difference of the average traffic throughput variation of the server-to-exit flow and the entry-node-to-client traffic (for all the clients). From our experience, we saw that for the victim traffic, the difference of the averages is within 120 Kbit/s. We used this as a threshold to eliminate flows whose average throughput, for the duration of the experiment, differed from the average throughput of the server-to-exit traffic by more than 120 Kbit/s.

These experiments, involving the server injecting the "square-wave" like pattern, were repeated 45 times (15 times corresponding to each client location). Average correlation between the server-to-exit traffic and entry node-to-victim client traffic was 0.57 (σ:0.22),



(a) Square-wave pattern (before rectification)

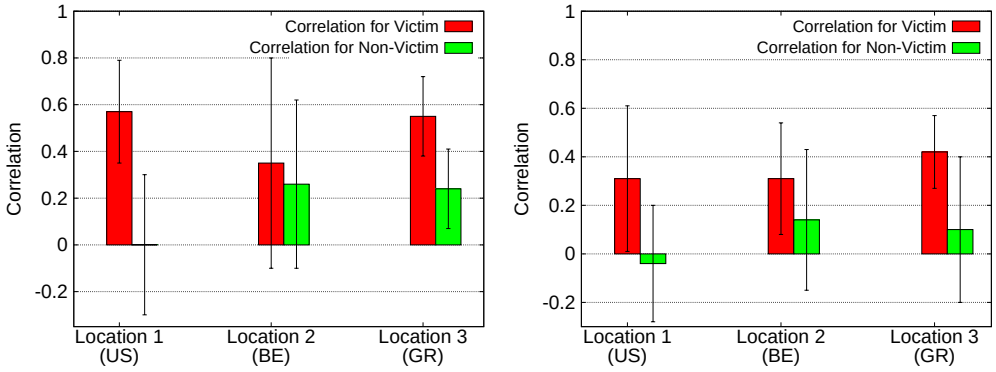
(b) Square-wave pattern (after rectification)

Figure 4.9: ((a) Server induced "square-wave" pattern of amplitude 1 Mbit/s along with other non-victim flows from the entry-to-victim and non-victim hosts having the four highest correlation co-efficient. Victim location : Texas, US. (b) Flows in Figure 4.9(a) adjusted and corrected using our rectification strategy.

0.35 (σ:0.45) and 0.55 (σ:0.17) for each of the clients at Texas (US), Leuven(Belgium) and Corfu(Greece) respectively (see Figure 4.10(a)). These averages were calculated from the flows having the highest correlation coefficients. The average of second-highest correlation coefficients, corresponding to non-victims, were 0.005 (σ:0.30), 0.26 (σ:0.36) and 0.24 (σ:0.17), respectively for each of the clients in Texas (US), Leuven(Belgium) and Corfu(Greece), corresponding to the three sets of experiments.

Similar experiments were also conducted with the server injecting a "step" like pattern obtained, by switching the traffic through 1 Mbit/s, 50 Kbit/s, 300 Kbit/s and 100 Kbit/s, every 30 seconds. Average correlation between the server-to-exit traffic and entry node-to-client traffic was 0.31 (σ:0.30), 0.31 (σ:0.23) and 0.42 (σ:0.15) for each of the clients at Texas (US), Leuven(Belgium) and Corfu(Greece) respectively (see Figure 4.10(b)). The second-highest correlation coefficients, corresponding to non-victims, were -0.04 (σ:0.24), 0.14 (σ:0.29) and 0.10 (σ:0.30), respectively for each of the clients for the three sets of experiments.

Overall we gathered a total of 90 measurement (thirty measurements for each of the three planetlab client locations). In 71 of those, we were able to correctly identify the victim flow (success rate of 78.9%). In 13 of the remaining cases we were not able to



(a) Average correlation: Square-wave pattern

(b) Average correlation: Step pattern

Figure 4.10: (a)Average Pearson's Correlation between server injected "square-wave" pattern and the victim and non-victim flows, for the different planetlab client locations. (b) Average Pearson's Correlation between server injected "step" like pattern and the victim and non-victim flows, for the different planetlab client locations.

correctly select the victim because either the correlation coefficient was either statistically not significant enough (< 0.2) or, the victim flow was filtered out because the average throughput differed from the average server-to-exit throughput by more than 120 Kbit/s.

There were six false positives in our measurements, where non-victim clients' traffic statistics were most correlated to the server-to-exit traffic statistics (carrying the injected pattern). Upon closer examination, we found about three instances where the number of sample intervals for the entry node to client was less than half the number of sample intervals corresponding to the server-to-exit traffic. These fewer sample intervals lead to loss of information and resulted in correlation representing an inaccurate relationship.

4.4.2.1 Monitoring multiple Tor relays

Finally, we evaluated our attack in a scenario involving an additional relay. We launched a second relay in our institution. The purpose of this second Tor relay was to judge the effectiveness of our attack in the presence more clients. On an average we saw about 900 users of our existing relay. The new relay attracted about 600 additional users.

We repeated some of the experiments we performed with the single Tor relay scenario.

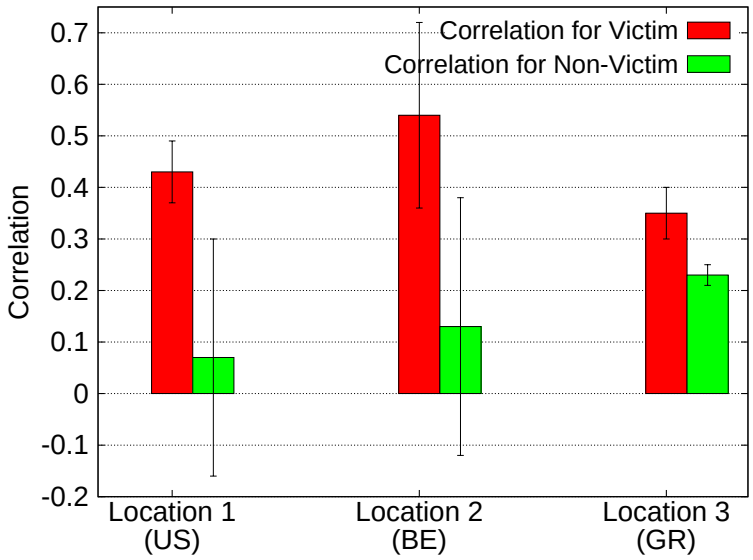


Figure 4.11: Average Pearson's Correlation between server injected "step" like pattern and victim and non-victim flows, corresponding to the different planetlab client locations, when monitoring the additional Tor relay. The entry node to client traffic data is captured using NetFlow data derived from institutional Cisco router with NetFlow capability.

In our experiments the server injected the complex "step" like pattern by switching the server-to-exit traffic between roughly 1 Mbit/s, 50 Kbit/s, 300 Kbit/s and 100 Kbit/s, every 30 seconds. These experiments were repeated 24 times, 8 times corresponding to each of the victim client locations. We observed higher average correlation between server-to-exit and entry-to-victim client traffic, compared to non-victim clients' traffic (see Figure 4.11). We were able to correctly identify the victim client in 14 out of the 24 trials (success rate 58.3%).

There were three false positives, where the correlation of the server-to-exit traffic was higher to a non-victim than to the victim. The remaining seven were false negatives, where the correlation coefficient was not significant (< 0.2).

This scenario, involving multiple relays, provides us with an intuition of what one can expect when an adversary monitors multiple relays. However, as mentioned in Section 6.1,

an adversary need not compare the server-to-exit flows to flows from all the entry nodes that it monitors. He (or she) could use traffic analysis methods [Chakravarty *et al.*, 2008b; Mittal *et al.*, 2011] to determine the actual entry node that is being used in the victim connection, thereby only using flow statistics corresponding to the victim entry node.

4.5 Discussions and Limitations

Our traffic analysis attack works well in a controlled in-lab set-up with symmetric network paths and path capacities (and other properties) with least congestion and external disturbances. In such an environment we were always able to determine the identity of the anonymous client amidst, several others clients that were communicating with the server. We observed sharp difference between the correlation of the server-to-exit traffic with that of the entry node-to-victim client and that between the server-to-exit node traffic with entry node-to-non-victim clients' traffic. This enabled easy detection of the victim traffic. This is evident from the results presented in the previous section. However, on moving to tests with real Tor relays, we did not see such high correlation for server-to-exit traffic and entry node-to-client traffic. This was because of existing path congestion and the traffic scheduling by Tor relays that distort the injected traffic pattern. The decrease in correlation is attributed to this distortion in injected pattern, thereby leading to information loss. In our experiments involving gathering of data from the institutional Cisco router, such effects were quite pronounced. Moreover, the number of sample intervals were lesser, compared to data obtained from Linux NetFlow packages. This was primarily due to flow aggregation, where multiple flow records are merged into a single records. It is quite likely that this arises due to increasing network load in the routers. This undocumented feature leads to to flow records that often have unequal lengths and are not evenly spaced. To compensate for such situations, we devised our approximation strategy that aligns server-to-exit and entry-to-client flow records and uses their rectified statistics as input to compute correlation coefficient. Such approximations can cause decrease in the overall correlation between the server-to-exit and entry node-to-victim traffic, since the process involves some information loss. It decreases the number of input points for correlation coefficient computation, result-

ing in a coefficient that doesn't accurately represent the variation of the server-to-exit and entry node-to-client traffic. For example, in several cases, even though the number of input points were more than 10 (often considered a minimum required number of samples to use correlation to demonstrate similarity in variation of the input samples). The false positive in our measurements were due of extreme lack of sample intervals, for the data captured between the entry node and the victim client (often much lower than the half the number of sample intervals for the data between the server and the exit node), even when they might be considered adequate enough to be used in correlation coefficient computation.

4.6 Conclusions

In this chapter, we presented a practical traffic analysis attack against Tor, that relies on using data from existing monitoring framework, already installed in network devices, such as Cisco's NetFlow. We have demonstrated the feasibility of launching an attack to determine the source of anonymous traffic. The idea of using NetFlow data to de-anonymize traffic was proposed earlier [Murdoch and Zieliński, 2007]. However, there the authors presented a theoretical model to study the attack and relied on simulation results. The focus had been to determine whether there were a small number of IXes from which such attack could be launched. We do not focus on finding appropriate vantage points and monitoring hosts, but rather on the logical "next-step", once such routers have been determined. We focus on studying how successful such an attack is in practice in identifying the source of anonymous traffic. We rely on correlation of traffic statistics to identify the source of anonymous traffic amidst various flows corresponding to clients using our entry node. Our research demonstrates such an attack first on an in-lab set-up involving a private Tor network and client which we controlled. In such an environment, free from external network congestion and artifacts related to link characteristics, we were able to determine the determine the actual source of anonymous traffic with 100% accuracy. Thereafter, we conducted experiments to test our accuracy in against real Tor traffic. In these experiments, involving planetlab clients and a public Tor relay (under our control), we were able to correctly identify the source of anonymous traffic in about 81.6% of our experiments (with about 5.5% false positives).

Chapter 5

Defending Against NetFlow Traffic Analysis Attacks

5.1 Overview

In the previous chapter, we described a traffic analysis attack involving data obtained from network monitoring infrastructure, installed in routers, such as Cisco's NetFlow. In our attack, we assumed that the adversary had access to flow records for traffic entering and leaving Tor entry and exit nodes that relayed the victim client's traffic. The adversary, correlating the statistics of traffic entering and leaving the Tor relays, could find similarities in network traffic, so as to link otherwise unrelated connections. In this chapter we propose some techniques to cope with such traffic analysis attacks. We propose a method to defend against the traffic correlation attack, that involves transmission of dummy traffic from the entry node to the victim client, consisting of packets with IP headers having small TTL values.

In the past, various methods to defend against traffic analysis attacks have been proposed, that involved the transmission of fake packets (also known as *cover traffic* or *link padding*) [Shmatikov and Wang, 2006; Fu *et al.*, 2003a; Wang *et al.*, 2008]. Some have also proposed deliberately dropping packets (called *defensive dropping*) [Levine *et al.*, 2004]. It is believed that such measures can potentially slow down the traffic flowing through low-latency anonymity networks, such as Tor. Only high-latency mix based systems (e.g.

Mixminion), can tolerate performance degradation due to such artificial delays. Such systems introduce artificial delays and re-order messages, with the sole purpose of defending against traffic analysis attacks that can correlate traffic entering and leaving relays and link apparently unrelated connections, thereby de-anonymizing anonymous connections. Traffic analysis by a powerful adversary thus continues to remain a threat in such systems because the various methods to defend against such attacks often degrade performance.

In the past, there have been several research efforts to use dummy packets judiciously for defending against traffic analysis attacks, without drastically degrading performance [Shmatikov and Wang, 2006; Wang *et al.*, 2008]. In these methods, the authors presented a dummy traffic sending scheme that observes the packets "on-the-fly" and estimates traffic characteristics such as inter-packet arrival times and throughput, and accordingly adjusts the rate of dummy traffic generation and transmission.

Our dummy traffic transmission mechanism has been designed to specifically to solve NetFlow traffic analysis attack described in the previous chapter. *Our motivation is to send dummy packets to distort flow records, without causing any significant effect on performance*. Our method relies on the entry node sending dummy traffic. These dummy packets appear identical to Tor packets (having the same IP header and packet length as that of regular Tor packets), but have IP headers with very small TTL values (generally 2 or 4). The small TTL values, causes them to be dropped within a few hops between the entry node and the victim client, thereby having less effect on client's performance. The fake packets travel along with the regular Tor packets but do not reach their destination. They are dropped soon after transiting the network edge routers. However, since they have the same IP address and port number values as that of regular Tor packets, they artificially distort the NetFlow statistics. Moreover, since they do not travel to their destination, they have very little to no effect on client-server traffic performance. Moreover, as described ahead in this chapter, we explored two dummy traffic transmission strategy wherein the entry node observes the outgoing traffic rate to determine when and at what rate to transmit dummy traffic. This idea has been inspired from the Adaptive Link Padding and Depending Link Padding strategies, which we described previously in chapter 2. The goal of these traffic transmission strategies is to reduce unnecessary transmissions (and thus the effect on user's

performance), while distorting the NetFlow statistics.

Our initial experiments involving transmission of such fake packets to defend against the NetFlow based traffic analysis attack, have yielded useful positive results. The correlation of the server-to-exit traffic, carrying the injected traffic pattern, with the entry node-to-victim traffic, is not higher than correlation of the server-to-exit and other non-victim clients' traffic.

Apart from the dummy packet transmission, in this chapter we also explore the possibilities of using Tor configuration parameters, originally designed for traffic shaping and conditioning, to potentially distort injected traffic pattern, thus thwarting the traffic analysis attacks. From our experience, we learnt that parameters designed specifically for traffic conditioning had no effect on the correlation of the server-to-exit and entry-to-victim client traffic statistics, and thus cannot be used to defend against our traffic analysis attack.

In the next section we describe in detail, our attacker model and our schemes to defend against such attacks, that involve dummy traffic transmission. Thereafter, we present our initial experimental results obtained by evaluating our defense strategies. Finally we conclude this chapter with a brief discussion regarding the results, highlighting the strategies that worked, and those that did not.

5.2 Attacker Model and the Approach Towards Defending Against Traffic Analysis Attack

In this section, we first discuss about the attacker model which we considered for this study. Thereafter, we describe our proposed methodology and experiments to defend against our NetFlow based traffic analysis attack.

5.2.1 Attacker Model

We assume the same adversarial model which we presented in the previous chapter. We assume a powerful adversary that can monitor traffic, through NetFlow statistics, at various network locations, allowing the inspection of Tor traffic towards a significant number of Tor nodes [Murdoch and Zieliński, 2007]. The adversary could identify important ASes

and their topological relationships using methods similar to those presented by Schuchard et al. [Schuchard *et al.*, 2012]. Alternatively, the adversary could use traffic analysis techniques to identify the entry node of a particular victim circuit [Chakravarty *et al.*, 2010; Mittal *et al.*, 2011]. Further, we assume that the victim connects to a server, that colludes with the adversary, and downloads a large file from the server. While the download progresses, the adversary injects a specific traffic perturbation pattern in the TCP connection it sees originating from an exit node. Thereafter, the adversary computes the correlation between the server-to-exit and entry-to-client traffic statistics, for all the clients (potential victims) and selects the one that is most correlated to the server-to-exit traffic.

The main objective of our attacker is to determine the source of anonymous connection arriving to a server using NetFlow data, available from routers (and from the malicious colluding server). The adversary uses statistical correlation to find maximum similarities between server-to-exit and entry node-to-client traffic, for all clients (one of which is the victim). It is well known that an adversary can compare traffic entering and leaving the Tor network to find network connections that show similar traffic patterns and link otherwise unrelated connections [Zander and Murdoch, 2008; Fu and Ling, 2009; Bauer *et al.*, 2007]. However, factors such as aggressive flow sampling by router and flow aggregation, coupled with bursty web traffic, network congestion and Tor's traffic scheduling, can often distort traffic patterns in TCP traffic transiting through the Tor network. We thus assume that victim downloads a relatively large file from the server, for the duration of the attack, so as to generate sustained traffic for a considerably long duration (atleast 5 – 7 minutes), and thus help generate adequate network statistics.

A victim could be lured to access a web-server through Tor. As mentioned previously, the potential victim can be enforced to download a file from a particular server in many ways. An adversary could, for instance inject (invisible) IFRAMEs in web sites frequented by the targeted victims, target particular victims using social engineering tactics (e.g. "spear-phishing" attack), or advertise decoy multimedia files, e.g., pirated movies. Further, injecting a specific traffic perturbation pattern, highlights the victim traffic amidst other contending flows, thus aiding in its identification.

In our experimental set-up, the victim client downloads a relatively large file (100

Mbytes) from the server, that colludes with the adversary, so as to inject a traffic pattern in the TCP connection it sees arising from an exit node. These traffic patterns are conceived by capping the connection throughput to a particular bandwidth value, for several seconds, and then switching to another one. We achieved this using Linux Traffic Controller [Hubert *et al.*,]. In our experiments, the adversary injected two different kinds of traffic patterns. The first one involves the server injecting a simple "square wave" like pattern, achieved by repeatedly switching the victim's traffic pattern between two bandwidth values. The other was a more complex "step" like pattern, that was achieved by the server switching repeatedly between several pre-determined bandwidth values.

After the download proceeds for a while, the victim connection is terminated and the traffic pattern injection is halted. The adversary obtains flow records from the server and the entry node (or router that had access to traffic going to and from the entry node), corresponding to the recently completed download. From these flow records, the adversary obtains statistics, namely bytes transferred, between the server and exit node and between the entry node and the various clients. The adversary then computes the correlation between the server-to-exit and each of the individual clients' statistics. The one that is most correlated to the server-to-exit traffic is chosen as the victim.

5.2.2 Defending Against Traffic Analysis Attack

We explored two different possible solutions to defend against the traffic analysis attack. The first one involved the entry node sending dummy traffic, so as to distort NetFlow statistics, and force the correlation analysis to present an inaccurate relationship between the server-to-exit and entry-to-victim client traffic. The second possible solution involves modifying the Tor configuration parameters for traffic shaping and conditioning, such as `BandwidthRate`, `BandwidthBurst`, `PerConnBWRate` and `PerConnBWBurst` and studying its effects on distorting the injected traffic pattern, enough to decrease its correlation to entry-to-victim client traffic statistics.

Defending Against Traffic Analysis Attack Using Dummy Traffic

As described previously, various mechanism have been proposed in the past, to defend against traffic analysis attacks. However, most of these proposals haven't yet been implemented because of possible performance degradation. We propose a method which uses dummy traffic to primarily defend against our attack involving NetFlow statistics. In our approach, the Tor entry node sends out dummy traffic which looks exactly like regular Tor cells but whose IP headers have very small TTL values (about 2 or 4). These dummy packets are sent out along with the regular Tor cells, so that they appear identical to the NetFlow monitoring sub-system, but they do not travel to the final destination, and are dropped within a few network hops along the path from the entry node to the client. Thus, these fake Tor cells, are not expected to cause significant slow-down of client-server traffic. We chose TTL value based on our set-up such that the packets leave the network perimeter of our institution, enough to distort NetFlow records at the edge router, but never travel further. An entry node operator can calculate the correct TTL value to use for their set-up (e.g. by using `'traceroute'`), prior to launching the relay service. This idea is schematically presented in Figure 5.1.

For our experimental evaluation, we used two dummy traffic sending strategies that could make it difficult for the adversary to determine the identity of the victim client by degrading the correlation of the server-to-exit and entry-to-victim client traffic. In both these strategies, the adversary observes the traffic throughput for the entry node to the victim client to observe when to decide when to inject the dummy traffic and when to stop it. These main objective of these dummy traffic transmission strategies is to optimize dummy traffic generation so as not to congest the victim client's connection. Both these strategies involve observing the entry node-to-victim client traffic to decide when and at what rate to transmit the fake packets.

These strategies are described as follows:

1. **Strategy 1:** The entry node monitors the traffic to obtain an initial estimate of the throughput values achieved by the client server traffic. Thereafter, whenever the traffic drops below a certain threshold (either deliberately due to the server injecting a

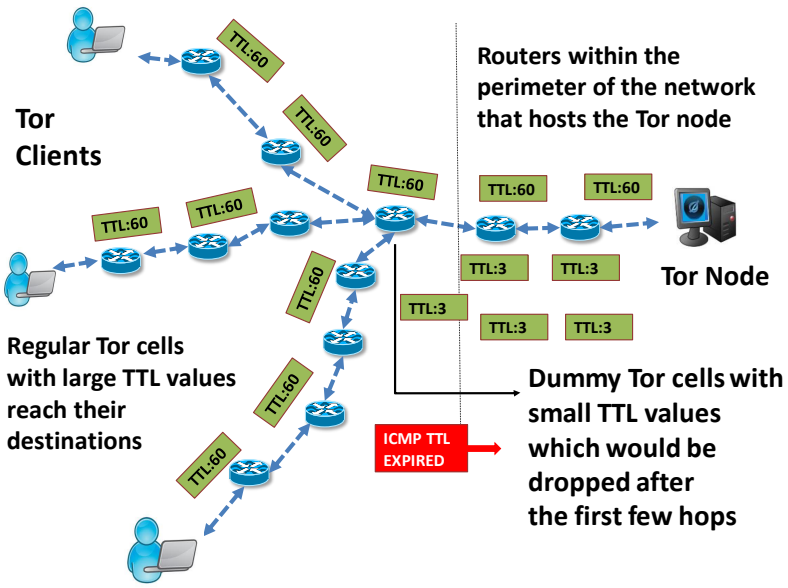


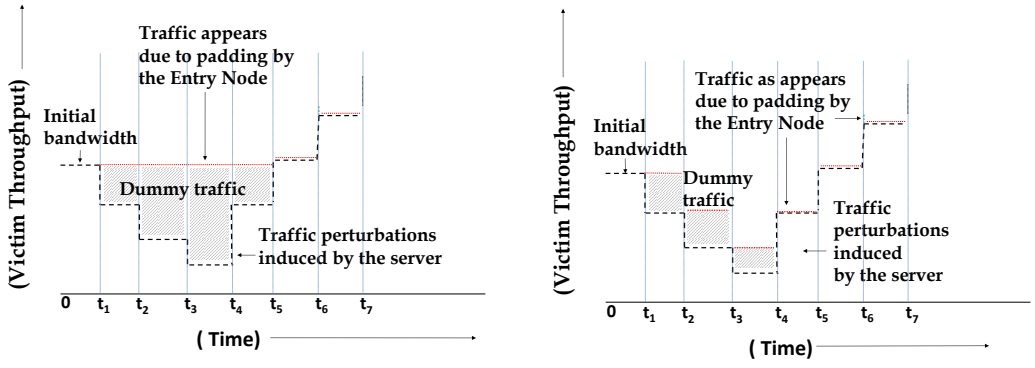
Figure 5.1: Dummy traffic with small TTL values that are dropped by the router beyond the edge of the network hosting the entry node.

traffic pattern, or otherwise due to network congestion and system disturbances)[1], the entry node injects the dummy traffic with the initially observed rate. Thus, an adversary observing the traffic between then entry node and the victim, never observes this decrease. To him (or her), it appears as if the traffic continues to flow at the initial rate. Again, when it rises above the previously recorded higher throughput, the server stops sending those dummy packets. Figure 5.2(a) pictorially demonstrates this traffic sending strategy.

2. **Strategy 2:** The second strategy is similar to the first one. The entry node measures the entry-to-victim client traffic throughput. Whenever it drops below a certain fraction of the initial bandwidth (say below 75% of the initially measured throughput), the entry node injects dummy traffic such that an adversary, observing the traffic, does not see this drop. The entry node continues to monitor the traffic and injects

[1]The entry node has no way to determine if the traffic perturbation was deliberately injected by a malicious server or a result of the prevailing network conditions

traffic when again the throughput drops, at a rate equal to the previous bandwidth rate. When the traffic rises above the previous high bandwidth level, the server stops sending the dummy traffic. Figure 5.2(b) pictorially demonstrates this traffic sending strategy. This strategy differs from the previous. In that the adversary observes no decrease at all in the entry-to-victim client traffic.



(a) Traffic padding strategy 1

(b) Traffic padding strategy 2

Figure 5.2: (a)Traffic padding strategy 1: Whenever the traffic drops (blow a certain threshold), pad the outbound traffic so that it appear that it is flowing at a constant rate (equal to some initial traffic rate).(b)Traffic padding strategy 2: Whenever the traffic drops below a certain threshold, pad the outbound traffic so that it appears as if it is flowing at a rate equal to the previous high bandwidth value.

One could conceive several other strategies. The experiments for evaluating the effectiveness of these traffic padding schemes is almost the same as those used to test the accuracy of our NetFlow traffic analysis attack. The client downloads a large file from the server, through a Tor circuit in which the entry node was the one that we controlled. Prior to the download, the client signals the entry node to protect its traffic against possible traffic analysis attacks. Thus the entry node monitors the victim's traffic periodically, depending upon the choice of strategy, to inject dummy traffic. After sometime, the download process is stopped and the entry node halts the dummy traffic sending procedure. Thereafter, much like the experiments presented in the previous chapter, the adversary correlates the server-

to-exit and entry-to-client traffic statistics, corresponding to all the clients that used the entry node during the experiment. The adversary selects the client that is most correlated to the server-to-exit traffic, amidst those entry-to-client flows where the average throughput of the experiments is comparable to the server-to-exit traffic, as the victim (as also done in the previous chapter).

Defence Using Traffic Shaping and Conditioning Parameters

Our second approach towards defending against traffic analysis attacks involves working with various Tor bandwidth shaping and traffic conditioning parameters. In our research we tried to to see if they are able to distort the injected traffic pattern and thus decrease the correlation between server-to-exit traffic and entry node-to-victim client traffic. These parameters have been designed keeping traffic shaping and conditioning in mind. Our approach is thus to see if we can use these parameters to distort deliberately injected traffic patterns and thus thwart traffic analysis attacks.

The are several built-in traffic shaping and conditioning parameters available in Tor. The following are the parameters whose effects we tried to explore through our research:

- `BandwidthRate:` The `'BandwidthRate'` parameter, limits the average incoming and outgoing bandwidth usage on the Tor node to the specified number of bytes per second. This is the maximum throughput any Tor circuit is allowed, when using this relay. The actual end-to-end transfer rate achieved by any circuit is however dependent upon the the `'BandwidthRate'` configured in each of the relays that is used by the circuit, the path bottleneck bandwidth and prevailing network congestion.

- `BandwidthBurst:` The `'BandwidthBurst'` parameter allows Tor to use more than the usual allowed bandwidth (`'BandwidthRate'`) when necessary, for very short periods of time. Using this parameter one can adjust the value of bandwidth burst for those short durations.

- `PerConnBWRate: 'PerConnBWRate'` allows the user to set separate `'BandwidthRate'` rate limitation for each connection from a non-relay (e.g. a client or a Tor bridge).

- `PerConnBWBurst`: Similar to the `'PerConnBWRate'`,`'PerConnBWBurst'` allows the user to set separate a `'BandwidthBurst'` rate limitation for each connection from a non-relay.

In our experiments, the victim performs its usual download from the server that modulates the server-to-exit traffic. We tested the effect of these parameters in an in-lab testbed. The above parameters were adjusted in all the relays. After a while, the download is stopped and the adversary tries to determine the correlation of the server-to-exit and entry-to-client traffic statistics (for all the clients), thereby selecting the client whose traffic statistics are most correlated to the server-to-exit traffic, as the victim.

5.3 Experimental Results

Having described our proposed approaches to defend against the NetFlow traffic analysis attack, we now describe our experimental efforts to test them, and the results obtained from those experiments. The first part of this section focuses on the experimental results we obtained in our initial efforts to explore the effectiveness of our dummy traffic sending scheme to defend against our traffic analysis attack. This was tested using our set-up involving a Tor client (hosted on a planetlab machine), our Tor entry node (that implemented the dummy traffic sending mechanism), and the server in Spain, that we controlled[2]. The experiments were essentially the same as those described in the previous chapter. The victim downloaded a large file from the server, that colluded with the adversary and injected a traffic perturbation pattern in the TCP connection, it saw arriving from an exit node. However, in these experiments, the entry node monitored the traffic to the victim client (that signals the entry node to defend its connection, prior to downloading the file), so as to inject the dummy traffic as and when needed. Thereafter, the adversary computes the correlation of the server-to-exit and entry-to-client traffic for all clients using the entry node for the duration of the experiment, and selects the client whose traffic is most correlated to the server-to-exit traffic, as the victim. Our initial experiments, involving the entry node sending dummy traffic Tor cells with small TTL values, that were dropped by routers within a

[2]This is the same sever to which the victim client connected in experiments described in the previous chapter

few network hops between the entry node and the victim client, yielded promising results in defending against the NetFlow traffic analysis attacks.

The second part of this section focuses on our efforts to explore the feasibility of using user configurable traffic shaping and conditioning capabilities built into Tor. This was tested through experiments conducted on a private Tor network (similar to the one shown in Figure 4.4). We tried to observe the correlation server-to-exit and entry-to-client traffic statistics for all the clients. During these experiments, we modified the configuration parameters of the relays. These modifications however showed no change in the correlation of the server-to-exit and entry-to-victim client traffic statistics, and the victim could be easily determined.

5.3.1 Defending Against NetFlow Traffic Analysis Using Dummy Traffic

We propose a technique to defend specifically against the NetFlow based traffic analysis attacks presented in the previous chapter. Our defense mechanism involves sending fake packets, that appears identical to Tor cells but whose IP headers have small TTL values. The TTL values are small enough such that these cells are dropped within a few hops along the path from the the entry node to the victim, and thus don't cause much congestion along the path from the entry node to the victim; while at the same time distort the entry-to-client traffic statistics recorded at the network edge router.

Our experimental evaluation of the defense technique involved experiments that were similar to the experiments described in the previous chapter. The experiments involved a planetlab client (hosted in Texas (US)) downloading a large file from the server (in Spain), using the entry node hosted in our University. Before initiating the download, the client signaled the entry node that it wishes its connection to be protected against traffic analysis attack. The entry node monitored the TCP connection from the victim client (using `'tcpstat'` [Herman,][3]) and injected dummy traffic as needed, depending upon the chosen link padding strategy. The flow records were generated and captured using the open source tools, running on the server and the entry node. While the client downloaded the file from the server, the server injected the traffic perturbation pattern into the TCP connec-

[3]a tool similar to `'tcpdump'` but focuses on presenting traffic statistics

tion, it saw arriving from an exit node. For our initial experiments, the server injected a "square-wave" like traffic pattern achieved by allowing the traffic to flow freely for about 20 seconds and then capping it to about 30 Kbit/s for the next 20 seconds. This generally resulted in a square-wave like pattern with fairly large amplitude (varying between 2 and 5 Mbit/s). After sometime, the experiment was terminated and the adversary obtained the flow records from the server, corresponding to the server-to-exit and entry-to-client traffic from the server and the entry node. The adversary calculated the differences of the average server-to-exit and entry-to-client traffic throughput and eliminated client flows whose differences from the server-to-exit flows exceeds 120 Kbit/s (as described in the previous chapter). Amongst the flows that remained, the adversary chose the one whose correlation to the server-to-exit traffic is the highest, as the victim client.

We conducted some initial experiments involving the two strategies. In all the experiments, involving both the strategies, the adversary failed to correctly identify the victim. The difference of the average traffic throughput of the server-to-exit and entry-to-victim client was higher than 120 Kbit/s in each of the cases, resulting in an incorrect identification of the victim, hence aiding the Tor client, that has chosen traffic padding based defense, to evade identification. The average correlation of the server-to-exit and entry-to-victim client traffic statistics, without any traffic padding defense was about 0.91 (σ:0.01). The average correlation of the clients' traffic, that were incorrectly identified as the victim and selected by the adversary, was 0.08 (σ:0.11) when using padding strategy 1, and 0.14 (σ:0.14) when using padding strategy 2. These results are summarized in Figure 5.3.

Figures 5.4(a) and (b) present sample flows for server-to-exit traffic carrying the injected "square-wave" pattern along with entry-to-victim client traffic that has been modified by injecting dummy traffic, generated using strategy 1 and 2, respectively. In each of these cases, the correlation of the server-to-exit and entry-to-victim client traffic was 0.07 and 0.1 respectively. In the first case, when the entry node sent dummy traffic using strategy 1, there were about 761 other clients simultaneously using the entry node, while in the second case, when the entry node sent dummy traffic using strategy 2, there were about 466 other clients (using the entry node). We did not observe any significant degradation of client's performance when dummy traffic was injected. Without the dummy traffic, the victim

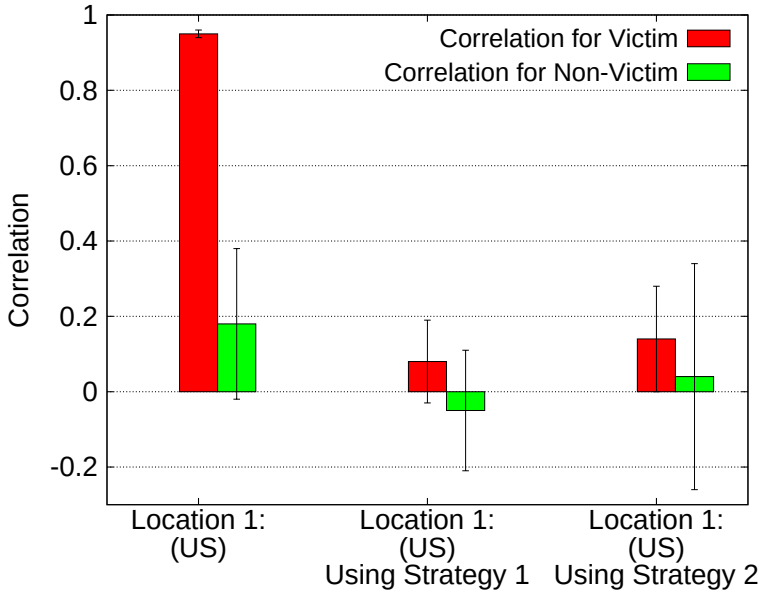
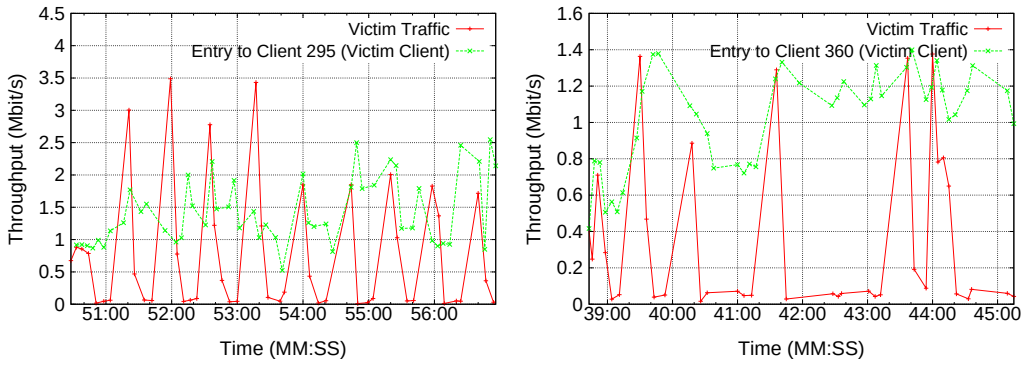


Figure 5.3: Correlation of server-to-exit and entry-to-victim client, with and without the defense strategies

client achieved a maximum of approximately 4 Mbit/s throughput when downloading the file from the server. Upon injecting the dummy traffic, the victim client's throughput did not vary significantly. There are two main reasons for this. Firstly, the dummy packets, having IP headers with low TTL values, were dropped within the first few network hops along the path between the entry node and the client. These packets do not travel to their final destination and thus have no effect on the available bandwidth of the network link leading to the client (that often has lower bandwidth than the intermediate over-provisioned inter-router links). Secondly, the host that runs the process for the Tor entry node, is over-provisioned. It connects to the ISP using a 1 Gbit/s link, having approximately 800–900 Mbit/s spare capacity. Our Tor entry node dedicates a maximum of 100 Mbit/s for serving Tor clients. With so much available link bandwidth, the performance of the entry-to-victim client suffers very little due to the dummy traffic.



(a) Padding Using Strategy 1

(b) Padding Using Strategy 2

Figure 5.4: (a)Sample server-to-exit and entry-to-victim client flows with padding traffic, generated using strategy 1 and (b) using strategy 2.

5.3.2 Defending Against NetFlow Traffic Analysis Attack Using Tor Traffic Shaping and Conditioning Parameters

The second part of this section describes our efforts to defend against the NetFlow traffic analysis attack using traffic shaping and conditioning parameters that are already built into Tor. Our objective was to see if the traffic shaping and conditioning parameters distorted the server injected traffic pattern and prevented the adversary from correctly determining the identity of the victim. To test the effect of modifying traffic shaping parameters, we began by using the set-up shown in Figure 4.4. While the client downloads the large file from the server, the server injects a "complex" pattern achieved by switching the server-to-exit traffic between 2 Mbit/s, 1 Mbit/s, 256 Kbit/s and 512 Kbit/s, every 10 seconds. There were a total of 30 clients simultaneously communicating to the server via the in-lab Tor network. In these experiments we try to see if the correlation of the server-to-exit and entry-to-victim client traffic is less than what we experienced previously (approximately between 0.70 – 0.99) in similar in-lab experiments, described in the previous chapter.

Our experiments involved varying `'BandwidthBurst'` parameter, while keeping `'BandwidthRate'` fixed. We conducted three sets of experiments, for different `'BandwithRate'` values (8 Mbit/s, 16 Mbit/s and 24 Mbit/s). For each set, we varied the value of

'BandwidthBurst' values between 8 Mbit/s, 16 Mbit/s, 24 Mbit/s, 40 Mbit/s, 56 Mbit/s and 80 Mbit/s. The maximum difference of correlations of the server-to-exit traffic and entry-to-client traffic, corresponding to victim and non-victim traffic respectively, was generally greater than 1.0. The difference of the highest correlation coefficient (corresponding to the correlation of the server-to-exit and entry-to-victim client), was always over 0.25, making it easy to identify the victim amidst the several clients. The correlation of server-to-exit and entry-to-client traffic is generally over 0.8. We did not observe any obvious degradation of the server-to-exit and and entry-to-victim client traffic.

We repeated these experiments, modifying the parameters 'PerConnBWRate' and 'Per-ConnBWBurst'. Similar to experiments involving the variation of 'Bandwidth-Burst' and 'BandwidthRate' parameters, here also, we did not see enough variation in the correlation of the server-to-exit and entry-to-victim traffic. We performed our experiments keeping 'PerConnBWRate' fixed while varying the 'PerConnBWBurst'. However, here as well, the adversary could easily identify the victim with significantly high correlation coefficient values (between 0.7 and 0.9).

5.4 Discussion

We explored the feasibility and effectiveness of two mechanisms to defend against traffic analysis attacks. These defense strategies can be enabled by Tor relays (particularly the entry node, as it is the one that knows the source of anonymous traffic). Our experiments to test the effectiveness of these strategies were essentially the same as those presented in the previous chapter, that were used to test the effectiveness of our NetFlow based traffic analysis attack.

The first involves sending padding traffic, along with regular Tor traffic, that appears identical to regular Tor cells, but whose IP headers have small TTL values, that cause them to be dropped within few network hops between the entry node and the victim client. These cells can effectively help reduce the correlation of the server-to-exit traffic, carrying the deliberately injected traffic pattern, and the entry-to-victim client traffic, by artificially modifying the traffic statistics as recorded by NetFlow routers. We have performed some initial

tests using our set-up involving planetlab client, our institutional Tor entry node and a server in Spain, that we controlled. In these tests, the data was gathered using open source tools. In these initial tests, the entry node sent fake packets based on two different traffic transmission strategies. In all cases, corresponding to these strategies, adversary failed to correctly identify the victim in all the tests. The difference between average traffic throughput of the server-to-exit and the entry-to-victim traffic was not comparable ($>>$ 120 Kbit/s) and thus was filtered out by the victim client selection process. On careful examination of the experimental results corresponding to the entry-to-victim client we discovered that the correlation of the server-to-exit and entry-to-victim client was almost in all cases insignificant (< 0.2) due to the injection of the dummy traffic, that modified the flow statistics. As a part of our future work we intend to gather more experimental results, not only using our set-up involving the open-source packages but also from our institutional edge router, to support our claims.

These traffic injection strategies had no significant effect on the throughput the client achieved, when communicating to the server via the entry node. In these strategies, the dummy Tor cells do not reach their destination and they do not have much effect on the client-to-server traffic.

Apart from transmitting dummy traffic, we explored modifying Tor's built-in traffic shaping and conditioning parameters to see if it had any effect on the injected pattern. In these experiments, we modified the various parameters, namely `'BandwidthBurst'`, `'Bandwidth-Rate'`,`'PerConnBWRate'` and `'PerConnBWBurst'`, to see if they had any effect on the injected pattern, thus reducing the correlation of the server-to-exit and entry-to-victim client correlation. We tried various combinations with the parameters. This did not have any effect on the correlation of the victim traffic statistics to the server-to-exit traffic and the adversary could easily identify the victim amidst the several clients.

5.5 Conclusion

In this chapter we focused on evaluating two proposals to defend against the NetFlow based traffic analysis that we described in the previous chapter. The first proposal involves the

entry node sending fake traffic, that appears almost identical to the victim traffic, to be sent towards the victim client. These fake packets have IP headers with small TTL values, so that they dropped (due to TTL expiry) within a few hops between the entry node and the client, and do not reach the client. These packets modified the traffic statistics recorded by NetFlow. Since these packets do not reach their intended destination, and get dropped after traversing over provisioned links, they do not affect the performance of client-server traffic. We tested two traffic padding strategies that involved monitoring the entry node-to-victim client traffic throughput and appropriately injecting the fake packets in such a way that an adversary, observing the network traffic, is not able to correctly correlate the server-to-exit traffic, carrying the injected traffic pattern to the entry-to-victim client traffic. In experiments, involving data obtained through open-source NetFlow packages, our attack methodology failed to correctly identify the victim client.

The second traffic analysis defense proposal involved modifying the Tor traffic shaping and conditioning parameters to reduce fluctuations in traffic, and thus the correlation of server-to-exit and entry node-to-victim client traffic statistics. These efforts did not yield any beneficial results towards defending against the traffic analysis attack and the adversary could easily identify the victim client in almost all cases.

Chapter 6

Determining Potential Adversaries of Anonymity Networks: Eavesdrop Detection in Anonymity Networks

6.1 Overview

This final chapter of the thesis focuses on ways of detecting anonymity network nodes that eavesdrop on the network traffic and are potential traffic analysis adversaries. In the previous chapters of this thesis, we focused on exploring the capabilities of adversaries, who, having access to network statistics for traffic entering and leaving the anonymity network nodes could correlate them to identity the source of anonymous connection. Such traffic analysis attackers, are in a position to observe traffic entering and leaving Tor, could also be potential eavesdroppers themselves.*Therefore one way to identify potential traffic analysis attackers could involve identifying anonymity network nodes that eavesdrop on users' traffic.*

Several services and protocols rely on non-encrypted communication. Consequently, malicious users or organizations that have access to the network elements through which user traffic is routed, can eavesdrop and obtain sensitive data, such as user authentication credentials. This situation can potentially worsen when users employ proxy-based systems

to access the same services without using end-to-end encryption, as the number of hosts or nodes that can eavesdrop on their traffic increases. Various public and private networks may block access to social networking and other popular online services for various reasons. Under these conditions, users often resort to using distributed proxying systems to prevent their traffic from being filtered. They resort to such mechanisms so as to evade network traffic filtering based on source, destination, and content.

Anonymous communication systems like Tor, are popular examples of proxy-based systems, which enable users to hide their IP address from the services they use, and often employ encryption by design. The data packets are encrypted several times, so that if adversaries intercept the traffic en-route to the destination, they will not be able to determine the actual source or destination of the traffic. The process also aids in achieving confidentiality against eavesdropping adversaries who can observe the traffic and snoop out sensitive and reusable information such as user names, passwords, and HTTP cookies.

In such systems, however, the last node on an anonymous path (e.g. Tor exit nodes), can access the original message that is being transmitted to the intended recipient. Many users are not aware of this discrepancy between the anonymity and privacy guarantees offered by these systems, and the lack of end-to-end data confidentiality which is often mistakenly assumed. Disregarding the absence of end-to-end confidentiality, users often send sensitive information through these relays. Some of these relays, acting with malicious intent, may misuse sensitive user information such as user names and passwords, URLs to sensitive information, and HTTP session cookies. Thus, in exchange for anonymity, users place their trust in components of the anonymous communication system that could potentially abuse it. In all cases, user data at some point is available in their original form. McCoy et al. [Mccoy *et al.*, 2008] have shown that there are Tor exit nodes which indeed eavesdrop on the traffic flowing through them, abusing users' trust.

An obvious solution to such problems might involve sending traffic encrypted using SSL through relays. However, malicious relay operators can employ man-in-the-middle attacks and snoop on the traffic of even SSL-encrypted sessions [Team Furry,], and attacks of this kind have been observed in the Tor network [Malicious Tor Exits,].

Our approach for the detection of misbehaving Tor nodes involves the transmission of

decoy traffic that contains easily reusable and seemingly sensitive, yet fake information (such as fake plain-text user names and passwords), via all nodes of the anonymization network to decoy servers under our control. Each exit node is associated with a unique decoy that is transmitted through it. Exit nodes eavesdropping on user traffic may try to reuse this information and connect to our decoy servers, and could thus be easily determined by checking the server logs for unsolicited connection attempts. Since our system transmits a unique decoy through each exit node, it is easy to determine the exit node involved. In this chapter, we present the overall architecture of our eavesdrop detection system, which can be used to detect eavesdropping by untrusted nodes of various anonymization systems (and proxying systems in general). We have implemented our system for detecting eavesdropping by malicious Tor exit nodes. However, it could be easily adapted for other relay based anonymization networks as well (such as JAP [JAP,] and I2P [i2p,]).

The use of fake information, or *honeytokens* [Spitzner,], for detecting unauthorized access to sensitive data, has been explored previously for several applications related to network intrusion detection and misbehavior detection. However, there has not been adequate research done in using such information and systems to detect misbehavior by otherwise trusted nodes of anonymization enabling networks and systems. McCoy et al. [Mccoy *et al.*, 2008] were the first to explore the use of transmitting decoy TCP traffic through Tor nodes to see which nodes eavesdrop on them. Their approach required access to DNS traffic and could be trivially defeated by an adversary, by simply using appropriate command line options of tools such as `tcpdump`. Our approach, in contrast neither requires access to DNS traffic, nor relies on default behavior of traffic capture tools. We send innocuous-appearing TCP traffic through Tor exit nodes exposing easily reusable decoy information, and periodically check the server logs for subsequent unsolicited connection attempts. The decoy information being unique to each Tor exit helps in easily identifying the Tor exit node involved in the eavesdropping incident.

In this chapter we present our overall architecture of the system that we have proposed to determine eavesdropping by nodes of anonymization network. Initially, we developed out system to support transmission of fake IMAP and SMTP delivery protocol messages, exposing fake usernames and passwords. In our paper [Charavarty *et al.*, 2011], we presented

details of some eavesdropping incidents recorded by our initial deployment. Based on the activities of the adversaries, recorded in the initial deployment, we augmented and optimized our system. The augmentations include SSH and FTP honeypots to lure adversaries to connect to them so as to gather more information about them, a web-server serving "sensitive appearing" decoy files, bearing *beacons* [Bowen *et al.*, 2009a; Bowen *et al.*, 2010; Bowen *et al.*,] that would trigger scripts to help locate the adversaries and systems to help determine SSL MITM attacks that involves exposing SSL handshake messages to exit nodes and subsequently validating the server certificates. Apart from the above additions, we have also explored how our system could be used to detect eavesdropping involving HTTP cookie hijacking. As a proof of concept, we have deployed various decoy servers and honeypots and transmitted decoy traffic to these systems via all Tor exit nodes. In a little over thirty-two months of its deployment, our system detected 18 incidents of eavesdropping.

The remainder of this chapter focused towards describing the architecture of our eavesdrop detection infrastructure, details of the eavesdrop detection incidents and various efforts we undertook to detect more advanced forms of eavesdropping (such as HTTP cookie hijack and SSL MITM attacks).

6.2 Relevant Research

There has been little effort in detecting misbehaving overlay nodes of anonymity networks. In a work most closely related to ours, McCoy et al. [Mccoy *et al.*, 2008] attempted to detect eavesdropping on malicious Tor exit routers by taking advantage of the IP address resolution functionality of network traffic capturing tools. Packet sniffing tools such as `tcpdump`, are by default configured to resolve the IP addresses of the captured packets to their respective DNS names. Their system transmitted, via Tor exit nodes, TCP SYN packets destined to unused IP addresses in a block owned by the system's operator. When the packet capturing program attempted to resolve the IP address of a probe packet, it issued a DNS request to the authoritative DNS server. The system's operator had access to the traffic going to this authoritative DNS server. Thus, requests to this DNS server with the unused IP address were

an indication that probe packets had been intercepted by some packet capturing program, and could be traced back to the network host where they were captured. However, when capturing traffic on disk, `tcpdump` by default does not resolve any addresses; and in any case the eavesdropper can trivially disable this functionality, rendering the above technique ineffective.

In contrast to that work, our system does not require access to the DNS server traffic. We present an overall system architecture that can easily be adapted to detect eavesdropping by exit nodes of Tor, for various kinds of traffic. We employ decoy clients, that communicate via Tor circuits, to a decoy server under our control and expose easily reusable, sensitive appearing information, such as user credentials and URLs to sensitive appearing documents to exit nodes. These decoys are unique to each exit node. Periodically, the client and server logs are tallied for unsolicited connection attempts, that are marked as suspicious. The decoys associated with the connection attempts, unique to each exit node, aids their identification. Our system is flexible enough to be adapted to detect eavesdropping in various different protocols. As a proof of concept, we have deployed a system that is not only capable detecting eavesdropping of IMAP, SMTP and HTTP traffic and also has a SSH honeypot and decoy FTP and HTTP servers, presenting decoy documents. Additionally, we also explored detecting HTTP cookie hijack attacks, for traffic destined to social network sites, and HTTPS man-in-the-middle attacks, by malicious exit node operators.

6.3 System Architecture

In this section, we present the architecture of our traffic eavesdropping detection system that we have deployed for Tor. We describe the design of the decoy traffic transmission mechanism and the corresponding decoy services, as well as the approach we used for incident data collection and correlation.

6.3.1 Approach

Eavesdropping on network traffic is a passive operation without any directly observable effects. However, the fact that some traffic has been intercepted can be potentially inferred,

when a third party that should not have access to the intercepted data uses it. For example, an eavesdropper can steal user credentials for services that do not use application-layer encryption, such as user names and passwords for websites with poor user authentication implementations, or for servers that use clear-text sign-in protocols, such as FTP or IMAP. Thereafter, any attempt by the eavesdropper to access the user's account is an observable event. The problem with the latter lies in (a service) identifying whether the use of a set of credentials was made by a third party or the user.

Our approach is based on the assumption that an eavesdropper will use the intercepted data in some manner. We use two types of decoy data, that are not used in any other way, to determine with certainty that data was intercepted by a proxy. First, we use decoy authentication credentials to decoy services, essentially honeypots, under our control. The use of these credentials with our services at a later time is a clear indication that eavesdropping occurred. Second, we transmit URLs to decoy documents containing information of potential value, such as decoy `PayPal.com` accounts, and fake financial transactions including fake credit card information. Later downloads of these documents also indicate eavesdropping. Every decoy is uniquely transmitted through exactly one proxy, so that we can later associate its use with it.

We further exploit the fact that the eavesdropper will probably attempt to open these documents to collect further information. We use `D-Cubed` [Bowen *et al.*,] to automatically generate the documents and embed *beacons*, basically scripts, into them. These scripts get launched automatically, when the documents are viewed by the eavesdropper, connecting to a remote host under our control, and transmitting information such as the IP address of the host used to open these documents. This aids us in gathering more information about adversaries, e.g. their geographic location based on their IP address.

Figure 6.1 illustrates the overall design of our system when applied on the Tor network. A client under our control periodically connects through Tor to our decoy server, and transmits easily reusable clear-text information, such as user authentication credentials. As a result, such easily re-usable and potentially sensitive information is exposed to exit nodes of Tor circuits (and any other network entity between the exit node and the decoy server). Both the client and the server record detailed information about any attempted connection

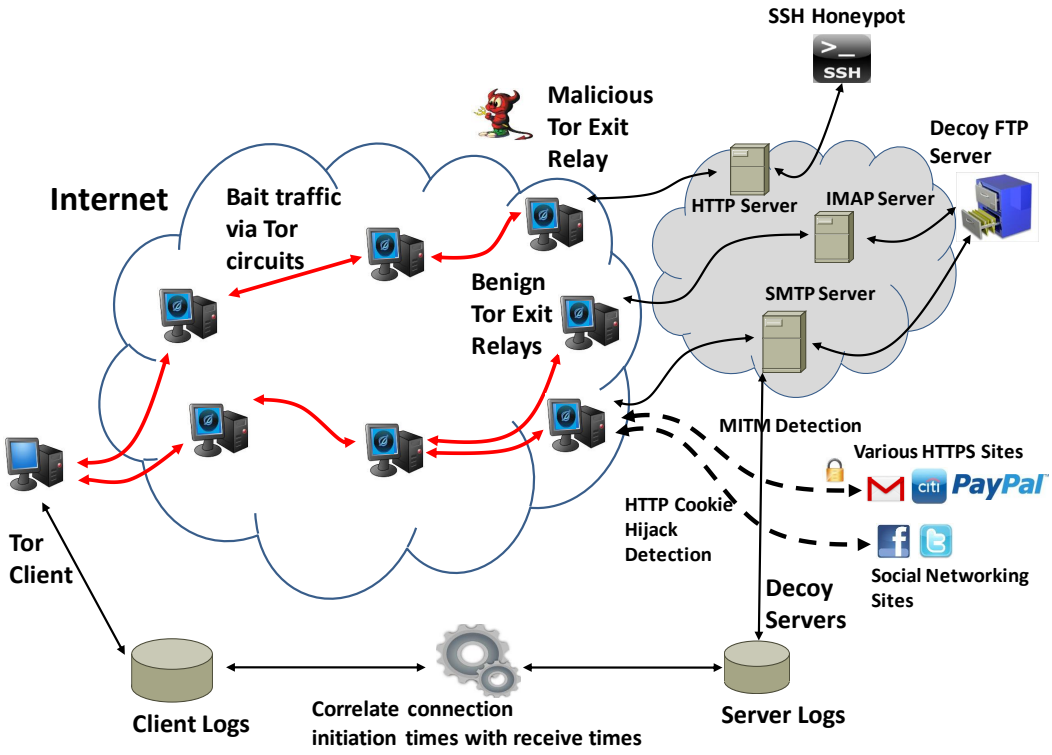


Figure 6.1: Overall architecture of the proposed traffic interception detection system when applied on the Tor network.

(such as user credentials, URLs, and connection timestamps). These logs are thereafter periodically tallied to determine unsolicited connection attempts, which are marked as suspicious.

In more detail, as the system is continuously running, the following steps take place periodically:

1. The client connects to the decoy server through Tor and sends information such as unique user authentication credentials (in case of IMAP and SMTP decoy servers) in clear-text, and URLs for sensitive appearing decoy documents containing beacons (in case of HTTP decoy server) through clear-text HTTP GET and POST protocol messages. The client creates circuits through *all* the exit nodes. Using unique user credentials and URLs per exit node helps in identifying the actual exit node involved

when eavesdropping is detected.

2. The decoy server maintains detailed record for each session, that may include the user name and password (for IMAP and SMTP), the IP address of the exit node used in the connection, the URL pointing to the unique decoy documents and the timestamp corresponding to when the connection to the decoy server was established.

3. After a successfully completed session on the decoy server, the system attempts to correlate it with a recently completed client session. Connections observed on the server for which there are no corresponding client connection attempts are labelled as suspicious.

Each of the unique user credentials and URLs to decoy documents is associated with exactly one exit node and is exposed to it through a Tor circuit terminating at that node. Thus, the exit node involved in a particular eavesdropping incident is known based on the given set of credentials or URLs used in the unsolicited session observed by the decoy server.

Our system has now been optimized compared to our initial prototype [Charavarty *et al.*, 2011] for gathering more information regarding the adversaries' activities. Attempts to reuse some of the IMAP decoy credentials with other services, like FTP and SSH, urged us to also set up honeypots with decoy FTP and SSH services running. Moreover, we used the FTP server to also serve decoy documents. We describe all incidents in detail, and present up-to-date information for new and previously detected eavesdroppers [Chakravarty *et al.*,], in Sec. 6.4.

Note that our approach can be adapted to detect more advanced traffic interception attacks. In Sec. 6.5 we describe an extension that can detect HTTP cookie hijacking attacks and man-in-the-middle attacks by malicious Tor exit nodes.

6.3.2 Implementation

Although Tor can forward the traffic of any TCP-based network service, in practice not all exit routers support all application protocols. For example, SMTP relay through port 25 is blocked by the majority of Tor exit nodes to prevent spammers from covertly relaying their

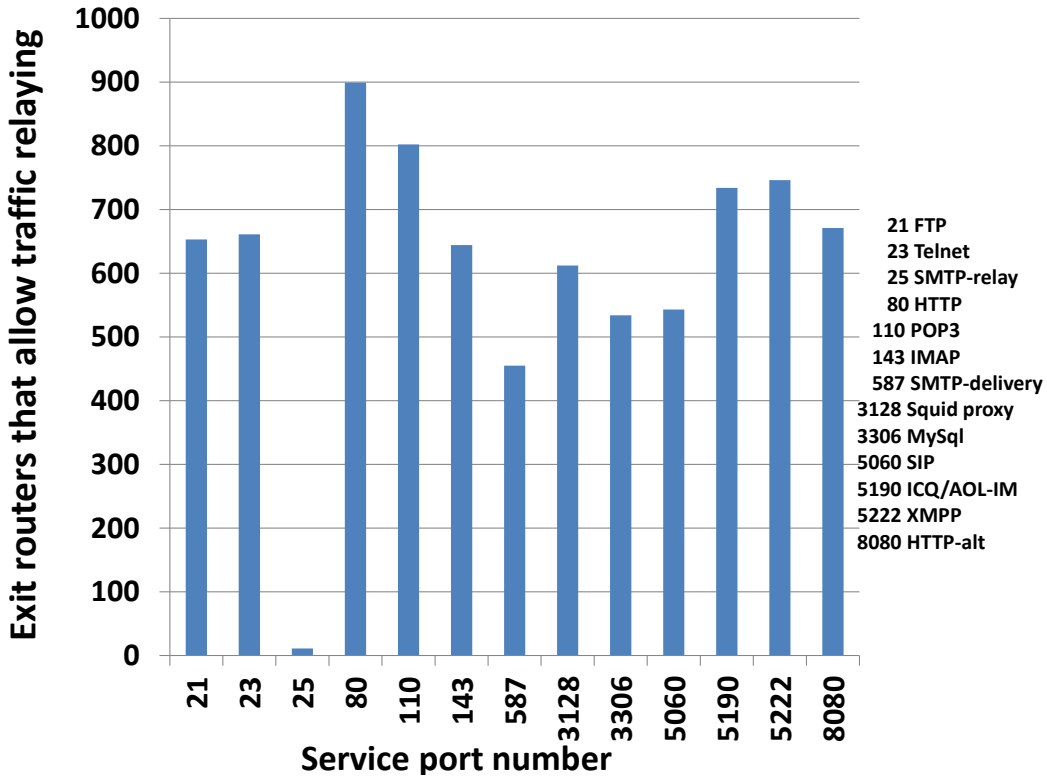


Figure 6.2: Number of Tor exit nodes that allow traffic relaying through different TCP port numbers, for services that support clear-text protocols.

messages through the Tor network. Consequently, the first important decision we had to take before beginning the implementation of our prototype system, was to choose a set of services that are supported by a large number of Tor exit nodes. At the same time, candidate services should support unencrypted authentication through a clear-text protocol, while the services themselves should be enticing for potential eavesdroppers.

Tor exit nodes are usually configured to allow traffic forwarding or only a small set of TCP services. The services allowed are defined by the operator of the exit node through the specification of an exit policy. To determine the most widely supported unencrypted application protocols, we queried the Tor directory servers and retrieved the number of exit nodes that allowed each different protocol. Figure 6.2 presents the number of Tor exit nodes that at the time of the experiment allowed the relaying of traffic through various TCP port

numbers. In accordance to the results obtained by McCoy et al. [Mccoy *et al.*, 2008], widely used applications like web browsing, email retrieval, and instant messaging are allowed by a large number of exit nodes. We found approximately 900 exit nodes that allowed access to port 80. We also found 644 exit nodes supporting exit to IMAP (port 143) and 455 exit nodes supporting SMTP delivery (port 587). Both these protocols support plain-text user authentication. They involve transmission of plain-text usernames and passwords.[1]

Credentials for accessing users' messages that may contain sensitive private information, or for sending emails through verified user addresses, can be of high value for a malicious eavesdropper. This led us to choose the IMAP and SMTP protocols for our prototype implementation. Furthermore, due to the wide prevalence of exit nodes that allow exit for web traffic, we decided to use HTTP GET and POST messages to expose URLs of decoy documents containing beacons, stored on a decoy web server that we control.

Decoy Traffic Transmission and Eavesdropping Detection

Our decoy traffic transmission subsystem is based on a custom client that supports the IMAP and SMTP protocols. The client has been implemented using Perl, and service protocol emulation is provided by the `Net::IMAPClient` and `Net::SMTP` modules. We use `curl` [Stenberg,] for transmitting the HTTP GET and POST messages to expose the URL of decoy documents to the exit nodes. The clients and servers are hosted on Intel x86 based machines running Ubuntu Linux.

Every day, for each service, our Tor client connects to the decoy service several times via circuits through each of the exit nodes. This is achieved by establishing a new Tor circuit for each connection, and enforcing each circuit to use a particular exit node. Once a connection has been established, the client authenticates on the server using a unique set of credentials tied to the particular combination of exit node and decoy server. Thereafter, the client performs activities, such as browsing through some folders in case of IMAP, or sending a fake email message in case of SMTP, so as the protocol message exchanges appear realistic. In case some exit node is not accessible, the corresponding set of credentials is

[1] In contrast to SMTP relay (port 25), SMTP through port 587 is dedicated to message submission for delivery only for users that have registered accounts on the server.

skipped. Similarly, when a new exit node joins the overlay network, a new set of credentials for each decoy service is generated for use only with that exit node. To achieve this, the Tor directory services are periodically queried and fresh list of exit nodes supporting exit to the requisite service (IMAP or SMTP delivery) is retrieved. Thereafter, new user credentials are assigned to the set of fresh exit nodes.

Usernames are generated as a combination of names in various languages [Pound, a] and random numbers using `language confluxer` [Pound, b] and `prop` [Pound, c]. Passwords for these usernames are generated using `pwgen` [Ts'o,].

Similar to the decoy IMAP and SMTP client processes, a routine sends and retrieves decoy documents to and from a decoy web server, through circuits via each Tor exit. The process exposes URLs of the decoy documents carrying the beacons, to the exit node through HTTP POST and GET messages. Each exit node is associated with a set of unique decoy documents which are sent to and received from the server, through HTTP POST and GET messages respectively. We assume rogue exit nodes to be snooping on HTTP traffic and accessing the exposed decoy document URLs.

Under normal operational conditions, the number of connections successfully initiated by the client each day through each exit node should equal to the number of connections received by the server from each of these exit nodes. Any unsolicited successful connection using some of the previously transmitted decoy credentials is labelled as an illegitimate suspicious connection attempt. Such suspicious connections are identified by tallying the connections initiated by our client to those received by the server, based on the logs recorded at the client and the server. Specifically, upon the completion of a successful connection, the decoy server sends directly (not through Tor) to the client all the recorded information about the recently completed session. The client then compares the connection details, including the set of credentials and decoy documents used and the start and end times of the connection recorded by both the client and the server, against the recently completed connections. In case no matching connection is found, the system generates a report that includes the time of the last generated connection that used the intercepted credentials, the time of the unsolicited connection to the server, the IP address of its initiator, and the exit node involved in the incident.

Important Implementation Considerations

During the implementation of our prototype system, we dealt with various issues related to improving the accuracy of our traffic interception detection approach, or with cases where interesting design trade-offs came up. We briefly describe some of these issues in the rest of this section.

Quality of Decoy Traffic and Honeypot Services The believability of the decoy traffic [Bowen *et al.*, 2010] is a crucial aspect of the effectiveness of our approach. For instance, a decoy IMAP session using an account that does not have a realistic folder structure, or that does not contain any real email messages, might raise suspicions to an eavesdropper. Repeating the same actions in every session, or launching new sessions at exactly the same time every day, can also be indications that the sessions are artificially generated. In our prototype system, we vary the connection times and activity in each session, we use realistically looking folder structures for the IMAP accounts, and send innocuous appearing email messages. The inboxes of these decoy accounts contain messages attached with decoy documents containing the beacons, generated from the `D-Cubed` system, presenting enticing information such as decoy `PayPal.com` accounts and fake financial transactions involving fake credit card numbers. These documents are attached to e-mail messages containing banking jargon to reduce suspicion.

As mentioned above, in some of the eavesdropping incidents, the adversaries actually tried to access other services such as SSH and FTP using the exposed IMAP credentials. Thus, we installed a FTP server, hosting user accounts corresponding to each of the IMAP users. Each of these accounts used the same passwords, which were used the IMAP accounts. The users' FTP directories were also populated with decoy documents containing the beacons. Further, to make these accounts appear innocuous, we also placed documents taken from [Services,] and source code documentations and help files taken from an open source program.

To track the behavior of adversaries who may try to log in to SSH accounts using the exposed IMAP credentials, we installed `kippo` [Desaster,], a medium-interaction SSH honeypot, on the virtual machine hosting our FTP server. These fake SSH user accounts

of `kippo` also used the same usernames credentials as those used for the IMAP and FTP accounts. The IMAP server redirects all FTP and SSH requests to this virtual machine (that hosts the SSH honeypot and the decoy FTP server).

Time Synchronization Accurate time synchronization between the client and the decoy server(s) ensures proper correlation of the connections generated by the client with the connections received by the server, and the correct identification of any unsolicited connections. Although the volume of our decoy connections is very low, allowing any illegitimate connections to easily stand out, the clocks of all hosts in our architecture are kept synchronized using the Network Time Protocol. The sub-second accuracy of NTP allows the precise correlation of the connection start and end times observed on both the client and server. This offers an additional safeguard for the verification of the detected traffic interception incidents.

Eavesdropping Incident Verification Besides the accurate correlation between the start and end times, logged by the client and the server, we have taken extra precautions to avoid any inaccurate classification of our generated decoy connections as illegitimate. For each connection launched by the client, the system also keeps track of the circuit establishment times by monitoring Tor client's control port. Moreover, we have enabled all the built-in logging mechanisms provided by the Tor software. On the server side, all the incoming and outgoing network traffic is captured using `tcpdump`. In addition to the server logs, the captured traffic provides valuable forensic information regarding the nature of illegitimate connections, such as the exact sequence of protocol messages sent by the attacker's IMAP, SMTP and HTTP clients.

6.4 Deployment Results

Our prototype implementation has been continuously operational in the Tor network since August 2010. During the course of over thirty months of its operation, our system has detected sixteen traffic interception incidents. In this section, we describe the eavesdropping and subsequent malicious connection attempts using the snooped user credentials. We an-

alyze the consequent activities of the intruders as they were recorded in the decoy server logs. Our incident description website [Chakravarty *et al.*,] contains information about the exit nodes involved in each incident, and details of the activities of the intruders once they logged in our decoy server using the snooped user credentials.

6.4.1 Eavesdropping Incidents

The observed eavesdropping incidents were related to different exit nodes, and all the related illegitimate connections were received by our decoy IMAP server. Based on the intercepted credentials used in each unsolicited connection, we were able to identify the Tor exit node involved in each incident. Information about the detected incidents (e.g., date, exit node location, and activities recorded by the server) are presented in Table 6.1. The detail of the remaining incidents are available in our website [Chakravarty *et al.*,].

While most of the incidents involved a different exit node, there were some, like one in India and another in South Korea, which eavesdropped on exposed credentials repeatedly and connected back to our decoy server. There were also several incidents where the malicious exit nodes were not accessible for days after the eavesdrop and subsequent connect back attempts. Also evident from Table 6.1, in the majority of the incidents the adversary connects to the decoy server via other exit nodes or hosts, in an attempt to hide his true identity. However, in the first four incidents, that occurred together, the connect back attempts originated directly from the exit nodes that were exposed to the decoy user credentials. All these connect back attempts occurred within four to six hours after the exposure of the decoy credentials to their respective exit nodes; which was significantly shorter compared to the rest of the incidents. We thus speculate that these first four eavesdropping cases were coordinated by the same individual or group, probably using the same tools or methodology in each case. Figure 6.3 presents this time differences between the exposure of the decoy credentials and the subsequent connect back attempts for each of the incidents. The horizontal axis represents the eavesdropping and connect back events. The vertical axis denotes the time delay between the exposure of the decoy credential and its subsequent usage in connect back attempts.

The map in Figure 6.4 presents an overall view of the geographic locations of the exit

Incident number	**Date**	**Exit node location**	**Remarks**
1	Aug.'10	US	Same pattern as in incidents 2, 3, and 4 Connect-back from the same exit node
2	Aug.'10	Hong Kong	Same pattern as in incidents 1, 3, and 4 Connect-back from the same exit node
3	Aug.'10	UK	Same pattern as in incidents 1, 2, and 4 Connect-back from the same exit node
4	Aug.'10	The Netherlands	Same pattern as in incidents 1, 2, and 3 Connect-back from the same exit node
5	Sep.'10	S. Korea	Connect-back from a different exit node
6	Sep.'10	Hong Kong	Connect-back from a third-party host Exit node not accessible upon detection
7	Sep.'10	India	Connect-back from third-party hosts Exit node not accessible upon detection
8	Jan.'11	Germany	Connect-back from third-party hosts Attempt to use SSL through the IMAP `STARTTLS` command
9	Apr.'11	India	Connect-back from third-party hosts and other Tor relays
10	Apr.'11	India	Same as 9. Both exit nodes in the same ISP network and many of the third-party connect-back hosts were in the same networks (mostly in Europe and India) Was involved in incident 7
11	Nov. '11	UK	Connect back via a host in a web-hosting and cloud service providing organization
12	Nov. '11	Russia	Connect back via another host in a Russian ISP
13	Nov. '11	S. Korea	Exit node involved in incident 5
14	Jan. '12	Germany	Connect back via another Tor exit in The Netherlands
15	Jan. '12	The Netherlands	Connect back via another Tor exit in Sweden
16	Jan. '12	US	Connect back via another host in a Canadian ISP

Table 6.1: Observed traffic interception incidents during first 19 months of the deployment. In all cases, the eavesdropper connected to our decoy IMAP server using a set of intercepted decoy credentials.

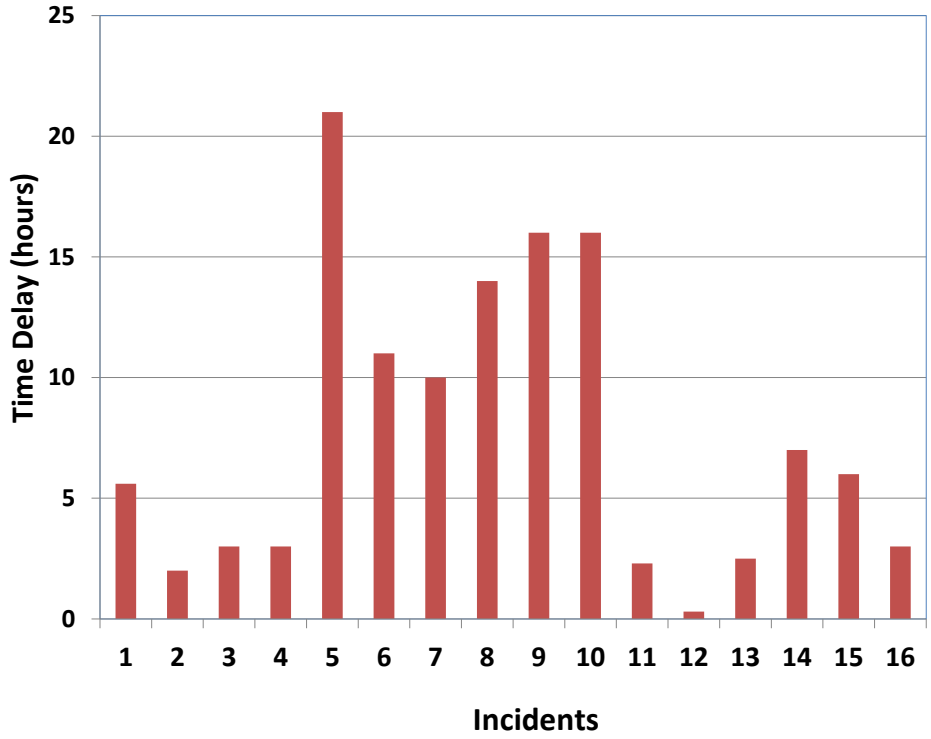


Figure 6.3: Time difference between the exposure of the decoy credentials and the first connect-back attempt on the decoy server.

nodes and the third-party hosts involved in the observed incidents. Tor and non-Tor nodes are represented using different symbols. We used basic geo-IP address lookup tools which provide only country-level accuracy, so the points on the map denote only the country in which each host was located. The number next to each point corresponds to the incident number, as presented in Table 6.1.

6.4.2 Adversaries' Activities

In some of the incidents, the adversary connected to the decoy server using popular e-mail clients; while in the rest, they connected directly to the server and manually issued protocol command messages. Popular email clients issue a certain default set of commands

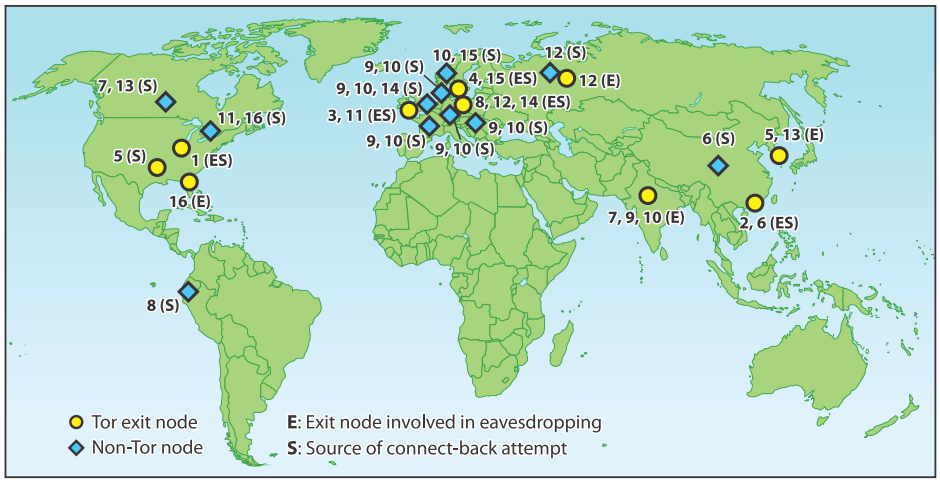


Figure 6.4: Locations of the Tor exit nodes involved in the observed traffic interception incidents, and the non-Tor hosts that connected back to the decoy servers. Numbers refer to the corresponding incidents listed in Table 6.1.

to access the mail folders such as `INBOX`, `Drafts`, `Sent`, and so on. The IMAP protocol allows users to issue various commands that can result in fetching the contents of these folders. Each mail clients issues a somewhat different set of commands to fetch the contents of these folders. The commands, and their order in which they are issued, can be treated as a "signature" for the client. We analyzed the signatures of various popular email clients (e.g., `Thunderbird`, `MS Outlook`, `Balsa` [balsa,], `Claw` [claw,], `Sylpheed` [sylpheed,], `Kmail` [kmail,] and `Evolution` [evolution,]) to determine the possible email clients the adversaries used. From the network traffic, corresponding to the time when the adversaries connected to the decoy server using IMAP clients, it appears that `Kmail`, `Evolution` and `Thunderbird` were commonly used for connecting to our decoy server.

In the incidents where the adversary connected manually to the IMAP server, we observed the adversary executing various different kinds of commands. In one, adversaries connected and switched to TLS mode so as to hide their activities. We thereafter turned off the capability in the server to switch to TLS mode after establishing connections. In another incident, the adversary issued esoteric IMAP4 ACL [Meyers,] commands. In yet

others, the adversary tried to use credentials to log into services such as FTP and SSH. These services were however inaccessible using the decoy user credentials. These activities compelled us to install the decoy FTP server and SSH honeypot. The IMAP server redirects the connection attempt to FTP and SSH services to the decoy FTP server and SSH honeypot so as to lure the attackers to the download decoy documents or try to execute programs from the terminal interface; thereby aiding in gathering further information about such attackers.

6.4.3 Volume of Decoy Traffic Injected

For each connection to the IMAP decoy server and subsequent protocol message exchanges, we sent approximately 15.4 KB of traffic. For 644 decoy accounts, each corresponding to a unique exit node, this number totals to approximately 10 MB. In case of such connections and exchanges to the SMTP decoy server, we sent about 21.3 KB of traffic. For 455 decoy SMTP accounts, this figure comes out to be approximately 9.7 MB.[2]

6.4.4 Attributes of the Exit Nodes Involved in the Incidents

Table 6.2 shows the available bandwidth of the exit nodes that were involved in the detected incidents. Two of the exit nodes advertised very high available bandwidth (44 and 20.8 Mbit/s, respectively), and thus are very likely to be selected in Tor client circuits, as the default Tor circuit node selection mechanism is biased towards nodes with high advertised available bandwidth [Tor Path Specifications,]. There were some which advertised somewhat lesser bandwidth of 8.5 Mbit/s and 1.4 Mbit/s. Finally, there were some which advertised yet lower bandwidths of less than 1 Mbit/s. Both of the high bandwidth, and two of the lower bandwidth nodes were *guard nodes*[3] with high up-times.

[2]This difference is primarily due to the different lengths of IMAP and SMTP messages. The overhead due to Tor protocol messages, involving circuit set-up, key exchanges, accounting, and circuit termination does not vary significantly between IMAP and SMTP.

[3]By default, a fixed set of entry nodes used by Tor clients to defend against traffic analysis attacks that can be launched by malicious entry and exit nodes

Incident number	Advertised Bandwidth	Remarks
1	Unknown	Relay was not running when accessed
2	Unknown	Relay was not running when accessed
3	44 Mbit/s	Guard node with high uptime
4	20.8 Mbit/s	Guard node with high uptime
5	1.4 Mbit/s	Advertises high uptime
6	56 Kbit/s	Advertises high uptime, runs directory service
7	Unknown	Relay was not running when accessed
8	856 Kbit/s	Guard node with high uptime, runs directory service
9	150 Kbit/s	Non-guard exit node
10	100 Kbit/s	Non-guard exit node
11	Unknown	Relay was not running when accessed
12	1 Mbit/s	Guard node with relatively high uptime
13	320 Kbit/s	Non-guard exit node
14	8.5 Mbit/s	Non-guard exit node with high bandwidth and uptime
15	Unknown	Relay was not running when accessed
16	336 Kbit/s	Non-guard exit node

Table 6.2: Available bandwidth of malicious exits (source: http://torstatus.blutmagie.de/)

6.5 Other Efforts and Possibilities: HTTP Session Cookie Hijack and SSL MITM detection

Apart from detecting eavesdropping on plain-text user credentials and HTTP URLs, our system is capable of being used for various other complex forms of traffic eavesdropping and misuse detection. As a proof of concept we tried to detect HTTP cookie hijack attacks and SSL man-in-the-middle attacks by malicious exit nodes. We elaborate more on these efforts in this section

6.5.1 Detection of HTTP Session Hijacking

Besides snooping on users' traffic, an adversary that has access to unencrypted network data can also mount HTTP session hijacking attacks against users that connect to social networking sites like `facebook.com`. Previously, such sites had no option to encrypt user

traffic except while authenticating them. Now, even when using HTTPS, there are various `facebook.com` applications that switch to HTTP and never switch back to HTTPS again, thereby exposing HTTP session cookies to eavesdroppers. In a session hijacking attack, the attacker can steal the session cookie that is included in the HTTP requests of authenticated users and use it to access the user's account. The fact that social networking sites are among the most frequently accessed websites Tor [Mulazzani *et al.*, 2010], combined with the ease of hijacking user sessions using tools like Firesheep [Butler,], makes the possibility of mounting session hijacking attacks on Tor exit nodes quite attractive for adversaries.

For detecting session cookie hijack attacks, we create several fake `facebook.com` user profiles. In this scheme, the decoy traffic consisted of activity generated by connecting to these `facebook.com` profiles and performing canned activities such as checking messages and status updates. However, we could not create unique `facebook.com` profiles for each of the approximately 900 Tor exit nodes that supported exit for web traffic. We thus created about twenty accounts and repeatedly logged into `facebook.com` by reusing the accounts. We exposed the first account in our list to the first exit node in our list, second account to the second one in our list and so on till the twentieth account. After logging in, our system checked the various user profile pages and private message folders for new messages. This process exposes the session cookies to the exit node several time for each of the accounts. Thereafter, our system, waited for a few minutes and checked the first twenty accounts for changes in profiles such as status update messages or private messages to others users in the contact list of the hijacked profile. Then, we again exposed the first user account in our list to the twenty first exit node, the second one to the twenty second and so on. We used a `firefox` browser automation framework, called `iMacros` [iOpus,], to perform these periodic logins and canned interactions. We ran our system for about six months but did not find any eavesdropping exit nodes sniffing on Facebook cookies.

6.5.2 SSL Man-In-The-Middle Attack Detection

Man-in-the-middle attacks have been observed by some malicious exit nodes [Malicious Tor Exits,] that try to intercept and compromise SSL key establishment process and use unverifiable or self-signed certificates. Our system can easily be adapted to detect such

man-in-the-middle attacks. To do so, we need to transmit SSL connections via Tor exit node to SSL services whose certificates can otherwise be verified. If in some cases we are unable to verify the server certificate when accessing the server via an exit node, we would conclude that the exit node has possibly manipulated the server to client traffic and might be intercepting the SSL connection establishment. In our set-up we used `curl` to connect to popular HTTPS sites such as popular webmail services and banks and checked if server certificate could be verified. One could use other tools such as [Palfrader,] for checking the certificates.

This process is repeated for all exit nodes. We found one exit node through which when SSL traffic was exposed, the server certificate verification failed. This node was however already already blacklisted in the Tor directory services and would not normally be selected when using the default client configuration.

6.6 Discussion and Future work

6.6.1 Detection Confidence

Internet traffic crosses multiple network elements until it reaches its final destination. The encrypted communication used in anonymity networks protects the original user traffic from eavesdropping by intermediate network elements, such as routers or wireless access points, until it reaches the boundary of the overlay network. However, the possibility of traffic interception is not eliminated, but is rather shifted to the network path between the exit node and the actual destination. Consequently, the transmitted decoy credentials in our proposed approach might not necessarily be snooped on the exit node of the overlay, but on any other network element towards the destination. This means that in the incidents detected by our system, the decoy credentials could have been intercepted at some other point in the network path between the exit node and the decoy server, and not at the exit node itself.

Although the above possibility can never be ruled out completely, we strongly believe that in all incidents the decoy credentials were indeed intercepted at the involved exit node for the following reasons. The ease of installing and operating a Tor exit node means that

not only adversaries can easily set up and operate rogue exit nodes, but also that exit nodes operated by honest individuals may be running on systems that lack the latest software patches, or have poor security configurations. This may enable adversaries to easily compromise them and misuse the hosted Tor exit node. At the same time, most of the network elements beyond a Tor exit node are under the control of ISPs or other organizations that have no incentive to blatantly misuse intercepted user credentials by directly attempting to access the user's accounts. Furthermore, in some of the cases, the adversary connected back to the decoy server from the same exit node involved in the particular eavesdropping incident, raising even more suspicion that the exit node is rogue or has been compromised.

Increasing Detection Confidence: Using Multiple Decoy Servers. As part of our future work, we plan to use multiple decoy servers scattered in different networks. Thereafter, we could check for eavesdropping and subsequent replay of user traffic, on each of the decoy servers. If eavesdropping is attempted on a traffic going to only a subset of the decoy servers, then it might be due to a malicious network router intercepting the path connecting the exit node to the said decoy servers. If however, eavesdropping is attempted for the traffic going to all the decoy servers via an exit node, it might have been perpetrated by the said exit node. Furthermore, one may use different sets of user accounts and decoy documents for the different decoy servers. Each of the exit nodes would thus be exposed to multiple sets of decoy user credentials, each one associated with a different decoy server. If, for a given exit node, eavesdropping is detected for one set of decoy user credentials or decoy documents, and not for others, then it might have been on network elements between the exit node and the corresponding decoy server, corresponding to the said set of decoy credentials or documents. However if eavesdropping is detected for all the decoy user credentials or documents exposed to it, it might likely be involving the exit node, because the network routers in the paths from the exit node to the individual decoy servers are exposed to different decoy user credentials. It seems less likely that a network router on a certain path would know the decoy user credentials that are exposed to routers on other network paths. That said, these measures do not completely rule out the scenarios wherein the adversary happens to eavesdrop on the traffic when the client access only one of the decoy servers and not the other(s) or when the network paths intervening the exit nodes and

the decoy servers intersect (and share common network elements).

6.6.2 Traffic Eavesdropping and Anonymity Degradation

Traffic eavesdropping on anonymous communication systems might not lead to direct degradation of network anonymity. However, inadvertently leaking user information such as login credentials can reveal vital information about the users, such as identity, location, service usage, social contacts, and so on. Specifically for Tor, the *anonymity set* commonly refers to all possible circuits that can be created, or the set of all possible active users of the system [Díaz *et al.*, 2003].

Traffic eavesdropping might help reveal information like the language and content of the messages, the particular dialect of the users, or other peculiarities that might help reducing the size of the anonymity set. For instance, a malicious exit node operator might see traffic carrying user data in Greek. Combined with the knowledge that there are about seven ISP networks in Greece, this information might help reducing the anonymity set significantly. Other clues such as the actual accessed content, the time of access, and the destination of the traffic, can as well aid the process of determining a user's identity.

6.7 Conclusion

Users of various anonymous communication networks, like Tor, often misconstrue the anonymity guarantees offered by such systems with end-to-end confidentiality. The use of encryption in systems like Tor, protect the confidentiality of the user traffic as it is being transported between the relays. This protects the original user traffic against surveillance by local adversaries, as for example in the case where the user is connected through an unsecured public wireless network. Even when encryption using SSL, users are not safe from man-in-the-middle attacks. In this chapter we have focused on the problem of detecting malicious eavesdropping nodes of anonymization networks like Tor. To tackle this problem we have presented an approach, involving the use of decoy network traffic injection, to detect rogue Tor exit nodes, engaged in traffic eavesdropping. Our approach is based on the injection of bait credentials and decoy documents, through Tor circuits, to decoy

services such as IMAP, SMTP and HTTP, with the aim to entice prospective snoopers to intercept and actually use the bait credentials and documents. The system can detect if a set of credentials has been intercepted, by monitoring for unsolicited connections to the decoy servers and by the alerts generated by the decoy documents containing the beacons, exposed to the exit nodes. We additionally run Honeypots to gather more information of the attacker. Moreover, our system can be easily adapted to detect more advanced traffic interception attacks such as HTTP cookie hijack and SSL man-in-the-middle attack.

Our prototype has been operational for over thirty-two months. During this period, the system detected eighteen incidents of traffic interception, involving exit nodes across the world. In all cases, the adversary attempted to take advantage of intercepted bait IMAP credentials by logging in on the decoy server, in some cases from the same exit node involved in the eavesdropping incident. Our system continues to run and detect eavesdropping by malicious exit node operators. Details of the latest incidents can be obtained from our website [Chakravarty *et al.*,].

Chapter 7

Lessons Learnt: Discussion, Limitations and Future Work

7.1 Discussions and Limitations

We began this thesis by discussing about events in the recent months pertaining to the disclosure of the mass surveillance activities by various government agencies and ISPs. To avoid such censorship, people have resorted to using censorship resistance and anonymity preserving systems, like Tor. We also mentioned how such agencies are seeking ways to censor and de-anonymize anonymous communication. However, as also mentioned, there have not been adequate efforts to fully evaluate the efficacy and effectiveness of attacks against anonymous communication systems, especially towards identifying the actual source of anonymous traffic. It is evident through past research efforts, that by and large, the process of de-anonymization can be divided into a two stage process. The first stage involves finding the set of relays and routers to monitor, that possibly forward the victim anonymous traffic. The second stage involves identifying the anonymous client, amidst network connections that use the reduced set of routers and monitors, generally through traffic analysis attacks, often relying on correlating variations in network statistics in different net-

work links. In the past, researchers have focused on the first phase of the attack, namely finding a small set of Tor relays or network routers that are likely relaying the victim's connection. One does not find adequate research efforts that involve identifying the anonymous victim within the anonymity set.

In our research we take steps to fill the gap. We studied two novel attacks that solely relied on network measurements, to aid the traffic analysis, that help verify the identity of anonymous connections. Additionally we also explored a method to defend against one of these attacks, that involved dummy traffic transmission in such a way that they do not congest the network connection between the entry node and the victim client and degrade the quality of service.

We began by exploring a traffic analysis attack, that relies on using remote network bandwidth estimation to confirm the identity of Tor relays and routers along a Tor circuit connecting a Tor client to the server. Initially we evaluated the effectiveness of our attack strategy in in-lab and DETER testbeds. In such scenarios we achieved 100% success in verifying the identity of relays and routers along the path to the victim. Thereafter we moved to experiments involving a Tor client, communicating to a server, under our control. The client communicated to the server via public Tor relays that served several other clients. The server, colluding with the adversary, deliberately injected traffic perturbations in the connection it saw originating from an exit node. This server was hosted on a node with high bandwidth (the node was hosted in a University and was connected to its network's edge router through a 100 Mbit/s link). The adversarial node tried to confirm the identity of the Tor relays and routers by detecting the deliberately injected bandwidth perturbations, using our single-end controlled bandwidth estimation tool (LinkWidth). We achieved moderate success in our efforts in confirming the identity of the Tor relays and network routers along the path between Tor entry nodes and the Tor client and Hidden Server.

In most of our experiments, involving the detection of traffic fluctuation on network routers and relays, the clients achieved low end-to-end bandwidth of approximately 300 Kbit/s. Detection of traffic patterns involving small fluctuations in high capacity links, in the presence of background Internet congestion, using probes, executed from non-vantage hosts, is prone to false negatives. However, adversaries equipped with a map of the ASes

intervening the path between different client subnets and the ASes hosting the Tor entry and exit node, such as those provided by services such as `iplane` [Madhyastha *et al.*, 2006] or `inano` [Madhyastha *et al.*, 2009], or through maps generation processes such as those highlighted by Schuchard et al. [Schuchard *et al.*, 2012] and Johnson et al. [Johnson *et al.*, 2013], and several high bandwidth monitoring nodes, could possibly traceback the server injected perturbations, accurately to the network hosting the anonymous client. Our attack strategy can be classified in the same category as those presented by Murdoch et al. [Murdoch and Danezis, 2005] and Mittal et al. [Mittal *et al.*, 2011]. Like theirs', our attack also relies on remotely measuring the variation of network statistics and correlating them to deliberately injecting fluctuation in anonymous client-server traffic statistics. Unlike theirs', our method doesn't rely on building single hop Tor circuits via candidate victim relays and can be used to confirm both relays and network routers that carry the victim traffic. Compared to our under-provisioned adversary, having scant vantage hosts, a powerful adversary, such as those described in the Introduction of the thesis, having several well provisioned vantage hosts, can measure changes to network traffic statistics and have better accuracy in confirming the identity of the victim.

The second traffic analysis attack assumes an adversary similar to the one assumed in the first attack. In the attacker model, the adversary colludes with the server to inject traffic perturbations in Tor connections to help identify the anonymous client that is communicating to the server through Tor, using NetFlow statistics. Murdoch and Zieliński [Murdoch and Zieliński, 2007], presented the first effort to demonstrate, albeit through simulations, that powerful adversaries, monitoring IXes, could use statistics from network infrastructure such as NetFlow and correlate traffic entering and leaving Tor entry and exit nodes, to determine the source of anonymous traffic. Explored purely through simulations, their paper does not present the reader any intuition of the accuracy of such attacks and technical challenges involved when practically executing them.

Our experiments, conducted to evaluate the traffic analysis attack, involved data collected from both open source NetFlow packages and from our University's edge router. Our correlation based attack strategy was tested with both adequate and sparse data. We have described the attack's accuracy obtained in identifying the victim client for both kinds of

data sources. We used a technique to compensate for sparse data points in NetFlow statistics obtained from our University's NetFlow router. In our experiments, although we used data obtained due to a single Tor entry node (with some experiments carried out with data from an additional entry node), we were moderately successful in identifying the victim amidst sever hundreds of competing clients (and in some cases over a thousand competing clients). These results provide the reader some information to speculate the accuracy of a powerful adversary, that is equipped to monitor several routers, to de-anonymize an anonymous connection and identify its actual source. However, our moderate success of identifying the source of anonymous traffic in about 81.4% of the cases, with about 6.4% false positives, is an indication that a powerful adversary, observing many more clients than we did, might not be able to accurately identify anonymous clients. We have some indications of degradation in accuracy with increase in the number of clients to monitor, through our experiments involving the second Tor relay that we launched in our University (see 4.4.2.1). Such degradation in accuracy is generally due lack of data samples in flow records gathered from Cisco's NetFlow framework, Internet congestion and Tor's traffic scheduling (issues which we touched upon earlier in chapter 4).

Alternately, a powerful adversary may employ attacks to identify the relays involved in a Tor circuit [Murdoch and Danezis, 2005; Chakravarty *et al.*, 2008b; Mittal *et al.*, 2011], and directly monitor traffic entering and leaving the entry and exit relays, to track down an anonymous client. Such efforts could be more effective than monitoring several network routers.

In chapter 5, we presented a novel dummy traffic transmission methodology to defend against our NetFlow based traffic analysis attack. The technique involves sending fake packets, alongside regular Tor cells, corresponding to the client that needs to be defended against traffic analysis attack. These packets have the same IP address and port number as the regular Tor cells, exchanged between the entry node and the client. However, their IP headers have very small TTL values. They are dropped soon after transiting the edge router of the network that hosts the entry node. The objective of our dummy traffic transmission strategy is to modify the NetFlow statistics so that the correlation of the server-to-exit traffic carrying the injected traffic pattern with entry-to-victim client traffic is low, and the adver-

sary fails to correctly identify the victim. At the same time, these dummy packets do not have much effect on the victim's performance as they are generally sent out, depending upon the observed entry node-to-victim client traffic throughput, through high provisioned network links between the entry node and network edge router, where they are eventually dropped. We have conducted some initial experiments involving our link padding strategy, using NetFlow statistics obtained from open source packages. In these experiments, the server-to-exit traffic is not well correlated to the entry-to-victim client traffic and thus is not selected as the victim. The entry-to-client traffic does not undergo any performance degradation. We have tried to conduct some experiments involving data obtained from our University edge router as well. However, due to extreme lack of sample points (often less than half the number of samples obtained for the server-to-exit traffic), made it hard to correlate the traffic statistics corresponding to the server-to-exit and entry-to-client traffic.

Finally, in chapter 6 we presented the architecture of a system which we designed and deployed to detect eavesdropping in Tor exit nodes. We began with a small set-up wherein we sent decoy IMAP and SMTP data, carrying unique, yet fake user credentials through Tor exit nodes to decoy server, under our control. Periodically client and server logs were tallied to find unsolicited connection attempts at the server, which were marked as malicious. Between August 2010 and March 2011 we detect about eight incidents of eavesdropping. Based on the activities of the various adversaries, observed by monitoring the traffic arriving to the decoy servers, we augmented our system with a medium interaction SSH Honeypot and a FTP server, to gather more information about the adversaries. The FTP server presented decoy documents containing scripts, called *beacons*, generated using the D^3 system [Bowen *et al.*,] system, that connect back to the D^3 server and report the IP address and the time when these documents are opened. We also deployed similar files on a HTTP server, and exposed the URL of the HTTP server to exit nodes through HTTP GET and POST messages. Further, we also tried to explore HTTP cookie hijack attacks in Tor exit nodes using a set of fake `facebook.com` and `twitter.com` profiles. At the time of the experiments, these systems used HTTPS for authentication but switched back to HTTP. We used canned page interaction using the `iMacros` [iOpus,] firefox plug-in used for automating browser actions (e.g. opening a web page, clicking specific links and

typing usernames and passwords) to expose the HTTP session cookies to the exit nodes and periodically checked the fake profile pages for changes to the wall and private messages. Further, we also tried to explore detecting simple forms of SSL MITM attacks by malicious exit node operators. To do so, a client accessed several HTTPS sites through the various exit nodes and checked for invalid server certificates. We found one exit node that possibly was launching such attacks. However, the exit node was already black listed by Tor Directory Services. Between August 2010 and March 2012, we detected a total of 18 eavesdropping incidents involving exit nodes that eavesdropped on decoy IMAP data were detected through their connect back attempts, using the exposed decoy credentials. The various augmentations did not detect any eavesdropping activities. Eavesdropping exit node operators could also be potential traffic analysis attackers. Thus, one could detect potential traffic analysis attackers by finding eavesdropping exit nodes.

7.2 Future Work

Having spoken briefly about outcomes the experimental efforts to study the traffic analysis attacks, the defenses and methods to determine eavesdrop by exit nodes, we briefly discuss about some of the future research efforts that one may pursue.

- **More measurements to validate the defense against NetFlow traffic analysis attack:** In this thesis we presented some initial measurements from evaluating our traffic analysis defense mechanism. We evaluated our defense mechanism against attacks involving flow records obtained from open source NetFlow packages, running on the server and the entry node. To obtain a better understanding of the effectiveness of the defense mechanism, and for the sake of comprehensiveness, we perhaps need to obtain more results from experiments involving flow records obtained from our University's edge router.

- **List of important ASes that can observe traffic to large fraction of Tor network:** Similar to efforts by Murdoch and Zieliński, one could identify a list of ASes, that lie at the intersection of AS paths connecting different client ASes and the ASes hosting Tor relays. Monitoring such ASes, can help the determine if there is a small set of

ASes that a powerful adversary needs to monitor to launch a traffic analysis, similar to ours. This could provide us the perspective of a powerful adversary, monitoring large fraction of Tor traffic through a small set of ASes, and de-anonymizing anonymous traffic using the NetFlow traffic analysis attack.

- **Multiple sets of decoy data to increase detection confidence:** As mentioned in chapter 6, in an attempt to make sure that traffic eavesdropping is conducted by exit nodes (and not by routers on the path connecting the exit node to the server), one could send multiple sets of decoy data through exit nodes. Each of the exit nodes would thus be exposed to multiple sets of decoy user credentials, each one associated with a different decoy server. If, for a given exit node, eavesdropping is detected for one set of decoy user credentials or decoy documents, and not for others, then it might have been due to network routers between the exit node and the corresponding decoy server, corresponding to the said set of decoy credentials or documents. However if eavesdropping is detected for all the decoy user credentials or documents exposed to it, it might likely be involving the exit node, because the network routers in the paths from the exit node to the individual decoy servers are exposed to different decoy user credentials. It seems less likely that a network router on a certain path would know the decoy user credentials that are exposed to routers on other network paths. Such measures have been described already been described in chapter 6.

- **More data to support accuracy of NetFlow traffic analysis attack:** In general, for having a better understanding of our NetFlow based traffic analysis attack, we need to gather data from multiple sources, involving data from multiple Tor entry nodes, at different geographic locations. This would provide us with a better understanding of the accuracy of our attack strategy.

Chapter 8

Conclusions

Although designed to enable low-latency anonymous communication system, systems like Tor, have served well as censorship circumvention tool. However, powerful government organizations, equipped with capabilities to monitor traffic in several networks (both local and international) and processing powers to mine and analyze network data corresponding to millions of users, have been seeking ways to de-anonymize and track down Tor users.

Systems like Tor are inherently vulnerable to traffic analysis attacks, wherein a powerful adversary, capable of observing traffic in several networks, can correlate statistics in them to find similarities and thus associate unrelated network connections. Such attacks can lead an adversary to the source of an anonymous connection. However practically locating the source of any anonymous connection could lead an adversary to search for patterns in almost all networks, across the globe. This being an impractical proposition, researchers have explored ways to reduce the set of network links and hosts to monitor, so as to identify the source of an anonymous connection. For almost seven years now, several researchers have explored various practical methods to reduce the of relays or network links to monitor. Having determined the set of relays or network links to monitor, the second part of such attacks involves identifying the victim amidst the network connections that are served by the reduced set of relays and network links to be monitored. Although, seemingly easier than monitoring all possible networks, there haven't been adequate efforts to explore the accuracy of traffic analysis attacks to identify an anonymous victim, amidst connections that are served by the reduced set of relays and network links. *Thus, opinions about anonymity*

and privacy guarantees of system such as Tor remain incomplete without the an adequate evaluation of all phases of the de-anonymization process.

To that end, we explored two novel active traffic analysis attacks that can be used to find an anonymous client within the reduced set of network connections. In both these attacks, the adversary colludes with the server to deliberately inject traffic perturbations in an anonymous connection and tracks them propagate through appropriate network links towards the source of the anonymous connection.

The first attack relies on our remote bandwidth estimation tool (called LinkWidth) to remotely measure change in available bandwidth, deliberately induced by the server, to observe them propagate through Tor relays and network routers connecting the client to the entry node. Our tool was moderately successful in confirming the identity of the Tor relays and routers intervening the path of the client and the server. The process of observing the fluctuations is similar to the ones presented in the effort by Murdoch and Danezis [Murdoch and Danezis, 2005] and by Mittal et al. [Mittal *et al.*, 2011]. However, unlike their approach, which could be used to only confirm the identity of the relays involved in the path, ours can be used to confirm the identity of both relays, as well as network routers.

The second attack uses correlation of NetFlow statistics corresponding to server-to-exit and entry-to-client traffic, corresponding to all clients, and selects the one showing the highest correlation as the victim. The traffic correlation attacks were evaluated through a set of experiments similar to those used to evaluate the accuracy of using LinkWidth to confirm the identity of the Tor relays and routers along the circuit connecting the client to the server. In controlled lab testbeds, we achieved 100% accuracy in identifying the anonymous victim. In tests involving data obtained due to circuits that used a public Tor relay, we were successful in about 81.4% cases to identify the victim amidst several hundreds of contending clients (with about 12.2% false negatives and about 6.4% false positives). We evaluated the tests with both adequate and sparse data sample points, obtained respectively from open source NetFlow packages, running on the server and the entry node, and from those using data from our University's edge router.

Having described this attack involving NetFlow based traffic correlation, in the fifth chapter we presented a methodology to defend against it. Transmitting dummy traffic to de-

fend against traffic analysis attacks, has been proposed several times in the past. None have however actually tried to implement and test it in the context of low-latency anonymous communication systems, because of possible performance degradation of users' traffic. We have presented a method to transmit dummy traffic that relies on sending packets with IP headers having small TTL values, that causes them to be dropped within few hops along the path from the entry node and the anonymous client. These packets, appearing identical to Tor packets, artificially distort flow statistics. At the same time, they have little effect on the client-server traffic performance, as they are dropped soon after crossing the edge of the network hosting the entry node. We have performed initial tests to evaluate the accuracy of this strategy to defend against the NetFlow traffic analysis attack. Our experiments showed promising initial results. The correlation of the server-to-exit and entry-to-victim client traffic decreased, making it difficult for the adversary to identify the victim.

We also tried to explore the effect of modifications of Tor's built-in traffic shaping parameters on distorting the server injected pattern and thus decreasing the correlation between the server-to-exit and entry-to-victim client traffic. Unfortunately, such modifications had no effect of diminishing the correlation.

Finally, we presented our system to detect eavesdropping by malicious Tor exit nodes. We began modestly by building a system to transmit fake IMAP and SMTP delivery messages via the exit node to a decoy server we controlled. Periodically the server and client logs were tallied to find unsolicited connection attempts at the server, that were marked as malicious. Based on some initial results, we augmented the system to add various components, such as Honeypots and decoy files, containing beacons, that would signal a remote server when accessed. We also explored if the system could be used to detect HTTP cookie hijack and SSL man-in-the-middle attacks. However, these efforts did not yield much results. Our system has detected 18 incidents of eavesdropping in over a period of thirty-two months.

Malicious eavesdropping exit nodes could also be party to traffic analysis attacks. Thus one way to determine potential traffic analysis attackers is to find eavesdroppers. Thus systems such as ours could be used to detect eavesdropping exit nodes, and thus potential traffic analysis attackers, that eavesdrop directly on traffic flowing out of the exit node.

All these efforts were conducted with our limited resources (consisting very few vantage points and relays to monitor traffic) and represents the capabilities of a weak adversary. In the past, Global Adversaries were merely considered hypothetical. Publications in the recent months [Schneier, 2013] have however confirmed otherwise. Powerful government organizations, such as NSA, seemed to have deployed Internet monitoring infrastructures to aid de-anonymization of anonymous communication. Based on our limited accuracy, one might speculate that for a powerful adversary, having visibility of large number of network connections might have lower accuracy in identifying the victim. However, such an adversary could employ traffic analysis techniques, such as the ones presented by Murdoch et al. [Murdoch and Danezis, 2005] or by Mittal et al. [Mittal *et al.*, 2011], to directly identify the relays involved in circuits and thus might not require monitoring traffic transiting large number of entry and exit nodes. Moreover, as described earlier, a powerful adversary equipped with several vantage nodes and high bandwidth may use such powers and employ our techniques to obtain higher accuracy in identifying the source of an anonymous traffic.

As anonymization systems such as Tor become vulnerable to traffic analysis attacks, users that solely seek censorship resistance should either use system such as Obfsproxy [Kadianakis,], that employ steganographic techniques to obfuscate anonymous communication or rely on new anti-censorship tools, based on *Decoy Routing* [Karlin *et al.*, 2011], such as Cirripede [Houmansadr *et al.*, 2011], that are resilient to censorship based on naïve traffic filtering, as they rely on network router based censorship resistance. Efforts to bypass such routers can cause several subnets to become unreachable and can also have economic impacts to the ISP that owns the router.

Bibliography

[Anonymizer,] Anonymizer, Inc. `http://www.anonymizer.com/`.

[Assange,] J. Assange. Wikileaks. `http://wikileaks.org/`.

[`balsa`,] Balsa - An e-mail client for GNOME. `http://balsa.gnome.org/`.

[Bauer *et al.*, 2007] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES)*, pages 11–20, 2007.

[Bauer *et al.*, 2008] K. Bauer, D. McCoy, D. Grunwald, and D. Sicker. BitBlender: Light-weight anonymity for bittorrent. In *Proceedings of the Workshop on Applications of Private and Anonymous Communications (AlPACa 2008)*. ACM, September 2008.

[Bennett and Grothoff,] K. Bennett and C. Grothoff. "gnunet:gnu's decentralized anonymous and censorship-resistant p2p framework". `http://gnunet.org/`.

[Bowen *et al.*,] B. M. Bowen, S. Hershkop, A. D. Keromytis, and S. J. Stolfo. D-cubed. `http://sneakers.cs.columbia.edu/ids/RUU/Dcubed/`.

[Bowen *et al.*, 2009a] B. M. Bowen, S. Hershkop, A. D. Keromytis, and S. J. Stolfo. Baiting Inside Attackers Using Decoy Documents. In *Proceedings of the 5th International ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, pages 51–70, September 2009.

[Bowen *et al.*, 2009b] B. M. Bowen, M. B. Salem, S. Hershkop, A. D. Keromytis, and S. J. Stolfo. Designing host and network sensors to mitigate the insider threat. *IEEE Security and Privacy*, 7:22–29, 2009.

[Bowen *et al.*, 2010] B. M. Bowen, V. P. Kemerlis, P. Prabhu, A. D. Keromytis, and S. J. Stolfo. Automating the injection of believable decoys to detect snooping. In *Proceedings of the third ACM Conference on Wireless Network Security (WiSec)*, pages 81–86, 2010.

[Burch and Cheswick, 2000] H. Burch and B. Cheswick. Tracing Anonymous Packets to Their Approximate Source. In *Proceedings of the* 14^{th} *USENIX Conference on System Administration (LISA)*, pages 319–328, December 2000.

[Butler,] E. Butler. Firesheep. `http://codebutler.com/firesheep`.

[cai,] CAIDA Router Measurements. `http://www.caida.org/tools/taxonomy/routing.xml`.

[CAIDA AS Peering Analysis,] CAIDA AS Peering Analysis. `http://www.netconfigs.com/general/anoverview.htm`.

[Chakravarty *et al.*,] S. Chakravarty, M. Polychronakis, G. Portokalidis, and A. D. Keromytis. Details of various eavesdropping incidents. `http://dph72nibstejmee4.onion/decoys_via_tor/map.html`.

[Chakravarty *et al.*, 2008a] S. Chakravarty, A. Stavrou, and A. D. Keromytis. Approximating a Global Passive Adversary Against Tor. Computer Science Department Technical Report (CUCS Tech Report) CUCS-038-08, Columbia University, August 2008.

[Chakravarty *et al.*, 2008b] S. Chakravarty, A. Stavrou, and A. D. Keromytis. Identifying Proxy Nodes in a Tor Anonymization Circuit. In *Proceedings of the* 2^{nd} *Workshop on Security and Privacy in Telecommunications and Information Systems (SePTIS)*, pages 633–639, December 2008.

[Chakravarty *et al.*, 2008c] S. Chakravarty, A. Stavrou, and A. D. Keromytis. LinkWidth: A Method to Measure Link Capacity and Available Bandwidth using Single-End Probes. Computer Science Department Technical Report (CUCS Tech Report) CUCS-002-08, Columbia University, January 2008.

[Chakravarty *et al.*, 2010] S. Chakravarty, A. Stavrou, and A. D. Keromytis. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In

Proceedings of the 15th European conference on Research in computer security, ESORICS'10, pages 249–267, Berlin, Heidelberg, 2010. Springer-Verlag.

[Charavarty *et al.*, 2011] S. Charavarty, G. Portokalidis, M. Polychronakis, and A. D. Keromytis. Detecting eavesdropping in tor using decoys. In *Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection*, pages 222–241, September 2011.

[Chaum, 1981] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.

[Claise,] Ed. B. Claise. Cisco systems netflow services export version 9. `http://www.ietf.org/rfc/rfc3954.txt`.

[`claw`,] Claws mail. `http://www.claws-mail.org`.

[Dabek *et al.*, 2004] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '04, pages 15–26, New York, NY, USA, 2004. ACM.

[Danezis *et al.*,] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: A Type III Anonymous Remailer. `http://mixminion.net/`.

[Desaster,] Desaster. `kippo` ssh honeypot. `http://code.google.com/p/kippo`.

[DETER Network Security Testbed,] DETER Network Security Testbed. `https://www.isi.deterlab.net`.

[Díaz *et al.*, 2003] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *Proceedings of the 2nd international conference on Privacy enhancing technologies*, PET'02, pages 54–68, Berlin, Heidelberg, 2003. Springer-Verlag.

[Dingledine *et al.*,] R. Dingledine, N. Mathewson, and P. Syverson. Onion Routing. `http://www.onion-router.net/`.

[Dingledine *et al.*, 2004] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium)*, pages 303–319, August 2004.

[Douceur, 2002] J. R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag.

[Edman and Syverson, 2009a] M. Edman and P. F. Syverson. AS-awareness in Tor path selection. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009*, pages 380–389. ACM, November 2009.

[Edman and Syverson, 2009b] M. Edman and P. F. Syverson. AS-awareness in Tor path selection. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009*, pages 380–389. ACM, November 2009.

[Evans *et al.*, 2009] N. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on tor using long paths. In *Proceedings of the* 18^{th} *USENIX Security Symposium (USENIX Security)*, pages 33–50, August 2009.

[Feamster and Dingledine, 2004] N. Feamster and R. Dingledine. Location Diversity in Anonymity Networks. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 66–76, October 2004.

[Flexible NetFlow Command Reference,] Flexible NetFlow Command Reference. `http://www.cisco.com/en/US/docs/ios/fnetflow/command/reference/fnf_cr_book.pdf`.

[Fu and Ling, 2009] X. Fu and Z. Ling. One cell is enough to break tor's anonymity. In *Proceedings of Black Hat Technical Security Conference*, pages 578–589, February 2009.

[Fu *et al.*, 2003a] X. Fu, B. Graham, R. Bettati, and W. Zhao. Analytical and empirical analysis of countermeasures to traffic analysis attacks. In *Proceedings of the 2003 International Conference on Parallel Processing*, pages 483–492, 2003.

[Fu *et al.*, 2003b] X. Fu, B. Graham, R. Bettati, W. Zhao, and D. Xuan. Analytical and empirical analysis of countermeasures to traffic analysis attacks. In *In: Proceedings of International Conference on Parallel Processing (ICPP). (2003*, pages 483–492, 2003.

[Gerla *et al.*, 2001] M. Gerla, M. Y. Sanadidi, R. Wang, and A. Zanella. TCP Westwood: Congestion Window Control Using Bandwidth Estimation. In *Proceedings of IEEE Global Communications Conference (Globecomm), Volume 3*, pages 1698–1702, November 2001.

[Goldschlag *et al.*, 1996] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of* 1^{st} *International Information Hiding Workshop (IHW)*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.

[Herman,] P. Herman. `tcpstat`. `http://www.frenchfries.net/paul/tcpstat/`.

[Honeynet,] The Honeynet Project. `http://www.honeynet.org/`.

[Hopper *et al.*, 2007] N. Hopper, E. Y. Vasserman, and E. Chan-Tin. How Much Anonymity does Network Latency Leak? In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, pages 82–91, October 2007.

[Hopper *et al.*, 2010] N. Hopper, E. Y. Vasserman, and E. Chan-Tin. How much anonymity does network latency leak? *ACM Transactions on Information and System Security*, 13(2), February 2010.

[Houmansadr *et al.*, 2011] A. Houmansadr, G.T.K. Nguyen, M. Caesar, and N. Borisov. Cirripede: circumvention infrastructure using router redirection with plausible deniability. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 187–200, New York, NY, USA, 2011. ACM.

[Hubert *et al.*,] B. Hubert, T. Graf, G. Maxwell, R. Mook, M. Oosterhout, P. Schroeder, J. Spaans, and P. Larroy. Linux Advanced Routing and Traffic Control HOWTO. `http://lartc.org/howto`.

[i2p,] I2P Anonymous Network. http://www.i2p2.de/.

[iOpus,] iOpus. imacros for firefox. `https://addons.mozilla.org/en-us/firefox/addon/imacros-for-firefox/`.

[Isdals *et al.*, 2010] T. Isdals, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving P2P data sharing with oneswarm. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 111–122, 2010.

[JAP,] JAP. `http://anon.inf.tu-dresden.de/`.

[`J-Flow`,] J-Flow Statistics. `http://www.juniper.net/techpubs/software/erx/junose82/swconfig-ip-services/html/ip-jflow-stats-config.html`.

[Johnson *et al.*, 2013] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users get routed: Traffic correlation on tor by realisitic adversaries. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS 2013)*, November 2013.

[Kadianakis,] G. Kadianakis. Obfsproxy. `https://www.torproject.org/projects/obfsproxy.html.en`.

[Karlin *et al.*, 2011] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer. Decoy routing: Toward unblockable internet communication. In *FOCI'11 - Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*, San Francisco, CA, USA, August 2011.

[Levine *et al.*, 2004] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright. Timing attacks in low-latency mix-based systems. In Ari Juels, editor, *Proceedings of Financial Cryptography (FC '04)*, pages 251–265. Springer-Verlag, LNCS 3110, February 2004.

[Madhyastha *et al.*, 2006] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. E. Anderson, A. Krishnamurthy, and A. Venkataramani. iplane: An information plane for distributed services. In *Proceedings of 7^{th} USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 367–380, November 2006.

[Madhyastha *et al.*, 2009] H. V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iplane nano: path prediction for peer-to-peer applications. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 137–152, Berkeley, CA, USA, 2009. USENIX Association.

[Malicious Tor Exits,] Known bad relays. `https://trac.torproject.org/projects/tor/wiki/doc/badRelays`.

[McCanne *et al.*,] S. McCanne, C. Leres, and V. Jacobson. `tcpdump` and `libpcap`. `http://www.tcpdump.org/`.

[Mccoy *et al.*, 2008] D. Mccoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker. Shining light in dark places: Understanding the tor network. In *Proceedings of the 8th international symposium on Privacy Enhancing Technologies (PETS)*, pages 63–76, 2008.

[Meyers,] J. Meyers. IMAP4 ACL extension. `http://www.ietf.org/rfc/rfc2086.txt`.

[Mittal *et al.*, 2011] Prateek Mittal, Ahmed Khurshid, Joshua Juen, Matthew Caesar, and Nikita Borisov. Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 215–226, New York, NY, USA, 2011. ACM.

[Mulazzani *et al.*, 2010] M. Mulazzani, M. Huber, and E. R. Weippl. Tor HTTP usage and information leakage. In *Proceedings of the IFIP Conference on Communications and Multimedia Security (CMS)*, pages 245–255, 2010.

[Murdoch and Danezis, 2005] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 183–195, May 2005.

[Murdoch and Zieliński, 2007] S. J. Murdoch and P. Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In *In Privacy Enhancing Technologies (PET), LNCS*. Springer, 2007.

[Murdoch, 2006] S. J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, pages 27–36, October 2006.

[National Security Agency/Central Security Service,] National Security Agency/Central Security Service. 'Tor Stinks' presentation. `http://www.theguardian.com/world/interactive/2013/oct/04/tor-stinks-nsa-presentation-document`.

[NetFlow iptables module,] NetFlow iptables module. `http://sourceforge.net/projects/ipt-netflow/`.

[`Netstream`,] Huawei NetStream Analysis, Monitoring and Reporting. `http://www.solarwinds.com/solutions/netstream-analyzer.aspx`.

[Øverlier and Syverson, 2006] Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2006.

[Palfrader,] P. Palfrader. Tor ssl mitm check. `http://svn.noreply.org/svn/weaselutils/trunk/tor-exit-ssl-check`.

[Pappas *et al.*, 2008] V. Pappas, E. Athanasopoulos, S. Ioannidis, and E. P. Markatos. Compromising Anonymity Using Packet Spinning. In *Proceedings of the* 11^{th} *Information Security Conference (ISC)*, pages 161–174, September 2008.

[Park and Crandall, 2010] J. C. Park and J. R. Crandall. Empirical study of a national-scale distributed intrusion detection system: Backbone-level filtering of html responses

in china. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 315–326, 2010.

[Pound, a] C. Pound. Chris Pound's Language Machines. `http://www.ruf.rice.edu/~pound/`.

[Pound, b] C. Pound. Language Confluxer. `http://www.ruf.rice.edu/~pound/new-lc/`.

[Pound, c] C. Pound. Prop. `http://www.ruf.rice.edu/~pound/prop`.

[Prasad *et al.*, 2003] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy. Bandwidth estimation: Metrics, measurement techniques, and tools. *IEEE Network*, 17:27–35, 2003.

[Price and Enayat, 2012] M. Price and M. Enayat. Persian cyberspace report: Internet blackouts accross iran;bbc journalists interrogated, family members imprisoned. `http://iranmediaresearch.com/en/blog/101/12/02/09/840`, February 2012.

[Provos, 2004] N. Provos. A virtual honeypot framework. In *Proceedings of the 13th USENIX Security Symposium*, pages 1–14, August 2004.

[Raymond, 2000] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[Raymond, 2001] J.-F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29, 2001.

[Reardon and Goldberg, 2009] J. Reardon and I. Goldberg. Improving tor using a tcp-over-dtls tunnel. In *Proceedings of* 18^{th} *USENIX Security Symposium 2009 (USENIX Security)*, August 2009.

[Reed *et al.*, 1998] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16:482–494, 1998.

[Reiter and Rubin, 1998] M. K. Reiter and A. D. Rubin. Crowds: anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1:66–92, November 1998.

[Risen, 2013] J. Risen. N.s.a. report outlined goals for more power. `http://www.nytimes.com/2013/11/23/us/politics/nsa-report-outlined-goals-for-more-power.html?pagewanted=all`, 2013.

[Rocketfuel,] Rocketfuel: An ISP Topology Mapping Engine. `http://www.cs.washington.edu/research/networking/rocketfuel/`.

[Ross, 2011] Quoted in "hilary clinton adviser compares internet to che guevara" by josh halliday. `http://www.guardian.co.uk/media/2011/jun/22/hillary-clinton-adviser-alec-ross`, June 2011.

[Schneier, 2013] B. Schneier. Attacking Tor: how the NSA targets users' online anonymity. `http://www.theguardian.com/world/2013/oct/04/tor-attacks-nsa-users-online-anonymity`, October 2013.

[Schuchard *et al.*, 2012] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper. Routing around decoys. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 85–96, 2012.

[Services,] Oxford University Computing Services. The university of oxford text archive. `http://ota.ahds.ac.uk/`.

[`sFlow`,] InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. `http://www.ietf.org/rfc/rfc3176.txt`.

[Shmatikov and Wang, 2006] V. Shmatikov and M. H. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Proceedings of ESORICS 2006*, September 2006.

[sidechan,] Side channel attack. http://en.wikipedia.org/wiki/Side_channel_attack.

[Spitzner,] L. Spitzner. Honeytokens: The Other Honeypot. http://www.symantec.com/connect/articles/honeytokens-other-honeypot.

[Spitzner, 2003] Lance Spitzner. Honeypots: Catching the insider threat. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC)*, 2003.

[Stenberg,] D. Stenberg. cURL. http://curl.haxx.se.

[Stoll, 1988] C. Stoll. Stalking the wily hacker. *Communications of the ACM*, 31(5):484–497, 1988.

[Tang and Goldberg, 2010] C. Tang and I. Goldberg. An improved algorithm for Tor circuit scheduling. In Angelos D. Keromytis and Vitaly Shmatikov, editors, *Proceedings of the 2010 ACM Conference on Computer and Communications Security (CCS 2010)*. ACM, October 2010.

[Team Furry,] Team Furry. Tor exit-node doing MITM attacks. http://www.teamfurry.com/wordpress/2007/11/20/tor-exit-node-doing-mitm-attacks/.

[The Internet Mapping Project,] The Internet Mapping Project. http://www.cheswick.com/ches/map/.

[Tirumala *et al.*, 1997] A. Tirumala, F. Qin, J. Dugan, J. Feguson, and K. Gibbs. IPERF. http://dast.nlanr.net/projects/Iperf/, 1997.

[Tor Metrics Portal,] Tor Metrics Portal. http://metrics.torproject.org/.

[Tor Path Specifications,] Tor Path Specification. https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt.

[Ts'o,] T. Ts'o. Password Generator. http://sourceforge.net/projects/pwgen/.

[evolution,] Evolution. http://projects.gnome.org/evolution.

[flow-tools,] flow-tools Package. http://freecode.com/projects/flow-tools.

[Fprobe,] NetFlow probes: fprobe and fprobe-ulog. http://fprobe.sourceforge.net/.

[kmail,] Kmail - mail client. http://kde.org/applications/internet/kmail.

[sylpheed,] Sylpheed-lightweight and user-friendly e-mail client. http://sylpheed.sraoss.jp/en.

[Wang *et al.*, 2008] W. Wang, M. Motani, and V. Srinivasan. Dependent link padding algorithms for low latency anonymity systems. In Paul Syverson, Somesh Jha, and Xiaolan Zhang, editors, *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, pages 323–332. ACM Press, October 2008.

[Wessels *et al.*,] D. Wessels, A.Rousskov, H.Nordstrom, A. Chadd, and A.Jeffries. Squid. http://www.squid-cache.org/.

[Winter and Lindskog, 2012] P. Winter and S. Lindskog. How the Great Firewall of China is blocking Tor. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI 2012)*, August 2012.

[Winter and Lindskog, 2014] P. Winter and S. Lindskog. Spoiled Onions: Exposing Malicious Tor Exit Relays. Technical report, Karlstad University, 2014.

[Wright *et al.*, 2002] M. K. Wright, M. Adler, B. N. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium (NDSS)*, 2002.

[Yuill *et al.*, 2004] J. Yuill, M. Zappe, D. Denning, and F. Feer. Honeyfiles: Deceptive Files for Intrusion Detection. In *Proceedings of the 2nd IEEE Workshop on Information Assurance (WIA)*, pages 116–122, 2004.

[Zander and Murdoch, 2008] S. Zander and S. Murdoch. An Improved Clock-skew Measurement Technique for Revealing Hidden Services. In *Proceedings of* 17^{th} *USENIX Security Symposium (USENIX Security)*, pages 211–225, San Jose, USA, July 2008.