ATTACKING ANONYMOUS COMMUNICATION NETWORKS THROUGH PORT REDIRECTION

BY

Muhammad Aliyu Sulaiman

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

May, 2012

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by *Muhammad Aliyu Sulaiman* under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirement for the degree of MASTER OF SCIENCE IN COMPUTER SCIENCE

Thesis Committee:

Dr. Sami Zhioua (*Thesis Advisor*)

Dr. Farag Azzedin (*Member*)

Dr. Jameleddine Hassine (*Member*)

Dr. Adel Fadhl Noor Ahmed

Department Chairman

Dr. Salam A. Zummo

Dean of Graduate Studies

16/7/12

Date

DEDICATION

I dedicate this thesis work to my Parents, my Wife, my little son Muhammad Kabeer, and all less privilege children whom because of war or famine where unable to attend schools.

ACKNOWLEDGEMENT

All praise be to Allah who thought by the use of Pen.

Acknowledgement is due to King Fahd University of Petroleum and Minerals for giving me opportunity to study and supporting this research.

My sincere appreciation goes to my Thesis Advisor, Dr. Sami Zhioua for opening my eyes to the world of Information Security and guiding me through this work. Equally, I would like to express my gratitude to the members of my thesis committee Dr. Farag Azzedin and Dr. Jameleddine Hassine for their time, guidance and suggestions.

I am thankful to the Dean of College of Computer Science and Engineering Dr. Umar Al-Turki and the Chairman of Information and Computer Science Department, Dr. Adel Fadhl Noor Ahmed and other faculty members for their cooperation and supports.

My acknowledgement goes to the members of my extended family; my parent, my wife and kid, my uncles, my aunts, brothers, sisters, friends and family friends for their unshakable caring feelings toward my studies and life in general.

I remain indebted to the members of the Nigeria Community here in KFUPM and elsewhere in the Kingdom for their fraternal support.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

THESIS ABSTRACT

NAME: MUHAMMAD ALIYU SULAIMAN

TITLE: ATTACKING ANONYMOUS COMMUNICATION NETWORKS THROUGH PORT REDIRECTION.

MAJOR FIELD: COMPUTER SCIENCE

DATE OF DEGREE: MAY, 2012.

This research is about investigating a new attack idea on anonymous communication systems, in particular Tor network. Tor is an open source anonymity network that helps users defends against a form of network surveillance that threatens personal privacy, confidential business activities and relationships. Tor is currently the state-of-the-art in low-latency anonymity systems and is the largest deployed anonymity network ever.

Since its first deployment in the late 2003, several vulnerabilities have been discovered and several attacks have been reported. At the same time, the design of Tor went through several modifications and improvements driven by the discovered vulnerabilities and the reported attacks. This research moves in the same direction as it investigates a relevant attack idea that would constitute a major threat to Tor clients. The attack aims to push the client to use specific exit policies (unpopular ports) for its traffic. This way, the path selection algorithm of Tor is manipulated to select among a small number of Tor relays from which the attacker controls an important fraction. The attack is successful if both the entry and exit nodes selected by the client happen to be compromised.

THESIS ABSTRACT (ARABIC)

ملخص الرسالة

الاسم: محمد علي سليمان

العنوان: مهاجمة أنظمة عدم الكشف عن الهويه من خلال إعادة توجيه المنافذ.

التخصص: علوم الحاسب الآلي

تاريخ التخرج: جمادى الآخرة، 1433 هجري.

هذا البحث يدور حول التحقيق في فكرة هجوم جديد على أنظمة عدم الكشف عن الهويه، بالخصوص على شبكة تور. تور هي شبكة مفتوحة المصدر تحمي المستخدمين ضد نموذج من مراقبة شبكة الاتصال تهدد الخصوصية الشخصية وأنشطة الأعمال التجارية السرية والعلاقات. تور حاليا هو من أحدث نظم عدم الكشف عن الهويه و هو أكبر شبكة متوفرة في أي وقت مضى. منذ الإعلان الأول عن هذه الشبكة و نشرها لأول مرة في أواخر عام 2003، تم اكتشاف العديد من الثغرات الأمنية وأفيد عن وقوع عدة هجمات. في الوقت نفسه، ذهب تصميم تور من خلال العديد من التعديلات والتحسينات مدفوعا بالثغرات الأمنية المكتشفة والهجمات التي تم الإبلاغ عنها. هذا البحث يتحرك في الاتجاه نفسه كما أنه يحقق في فكرة هجوم يشكل تهديدا كبيرا لعملاء تور. الهجوم يهدف لدفع العميل لاستخدام مدخل لا يحظى بشعبية و بهذه الطريقة ، يتم التلاعب ببرنامج اختيار المسار ليقع الإختيار على ملقم تحت سيطرة المهاجم.

CHAPTER 1

INTRODUCTION

As people are massively depending on Internet in the recent years for their day to day activities, privacy is undoubtedly a critical issue. Opinion poll keeps reminding us that one of the most important reservations most users of the Internet have, is the fear of having their privacy been infringed. Unfortunately, that isn't unjustifiable as corrupt marketers, private and government security agencies have been aggressive in monitoring and censoring user's activity on the Internet.

Today there are some compliant network services and useful data encryption algorithms to protect network traffic. However it is still hard to hide routing information such as source and destination addresses, packet length, etc. For this reason attackers can make use of such parameters to obtain some valuable information about users.

Anonymity System is an information security term that refers to a measure taken to protect user privacy over a network in which the identity of a sender (information source) and a recipient (information destination) is concealed from public including a legal network monitoring agency. More precisely, it is the state of being not identifiable within a set of subjects (that is anonymity set) [1]. At present various technologies of anonymity have been employed in various fields of human endeavor such as e-voting, e-commerce, e-banking, e-auction and many more.

Tor is undoubtedly the most popular low latency TCP based anonymity protocol that supports wide range of Internet applications such as web browsing (http), file transfer protocol (ftp), instant messaging (chat), file sharing and email clients. This research work focuses on Tor Networks.

1.1. MOTIVATION

Tor is the most widely used anonymity network all over the world. It has enabled approximately 36 million people around the world to experience fundamental freedom of access and expression on the Internet while protecting their privacy and anonymity [15].

Tor is an open source Protocol which makes it possible for researchers to mount several attacks on the protocol and exposing vulnerabilities with the intention of revealing possible flaws and provide mitigation or suggest solutions. This has led to several update

of Tor software. In the same spirit we studied Tor Protocol intensively, present a new attack scenario and provide mitigation.

1.2. BACKGROUND OF THE STUDY

Tor is an onion routing anonymity system originally developed for the U.S. Navy primarily to protect government communications. However, this purpose has dramatically changed after it has been donated to the Open source community. Tor is reported to have clients in more than 126 different countries with Germany, China, and United States as the major users of Tor [2].

Tor is used by people with varieties of purposes. Normal people use Tor to protect their privacy as well as that of their children from being stolen by corrupt marketers and identity thieves. Military use Tor to hide their location, protect military interests and operations, as well as protecting themselves from physical harm when deployed in the field. Journalists use Tor to investigate state propaganda and opposing viewpoints, to file uncensored stories and avoid the risk of being arrested. Activists use Tor to anonymously report abuses from danger zones of the world. Bloggers use Tor to avoid being sued or fired for saying completely legal things online. Law enforcement officers use Tor for carrying undetected surveillance on questionable web sites and much more [3]

Tor protocol utilizes the power of the two main anonymity approaches. Mixing and proxy, it uses public key cryptography to establish circuit and use symmetric key cryptography to move data like SSL-TLS based proxies, this makes it possible to fool Bob in believing that a request is coming from a routing relay and not Alice. In addition, the protocol distributes trust among the intermediary relays just like mix system, a condition that makes it difficult at least for any malicious relay node to know that the transmitting payload is channeling to Bob or the data is coming from Alice. Tor protocol system has four recognizable features:

I. Perfect forward anonymity, incrementally building circuit through session key negotiation with each next node and regular renewal of circuit.

II. Several TCP streams are multiplexed in a circuit, which helps in reducing latency.

III. Leak pipe circuit topology, Tor initiator is responsible for directing traffic to nodes pathway down the circuit, meaning that the traffic can exit the circuit at the middle thereby prevent end-to-end attack.

IV. Transaction between nodes is TLS based, which makes it difficult to modify the transmitting data and as well depict a relay.

1.2.1. Tor Unpopular Ports

Tor is mainly designed to support Internet applications which typically require high responsiveness and because of the fact that certain applications have tendency to leak some fragments of relay information or making Tor relays vulnerable to spams and viruses, Tor default exit policy rejects relay requests for some applications (ports) through Tor Networks as shown in table1.1, thus they are termed as unpopular ports.

TABLE **1.1**: List of Tor unpopular ports which are rejected by the default Tor exit policy

Port
25
119
135-139
445
563
1214
4661-4666
6346-6420
6699
6881-6999

1.3. PROBLEM DEFINITION

Typical attacks on Tor anonymity network aim to reveal the identities of clients (senders) and/or servers (destinations). However, Tor tries to hide the connection between clients and servers by forwarding the traffic through a circuit composed of typically three nodes. A successful attack involves identifying the first node in the circuit (entry-node) and the last one (exit-node). In a very simple attack, a malicious party may inject several Tor relays in the network and then waits until a client selects two of these compromised relays as entry and exit nodes. This will reveal both the client and the server identities. Since the path selection algorithm used by Tor is bandwidth-weighted (high bandwidth relays tend to be selected with high probability), an improved version of the attack would be to inject malicious Tor relays with high bandwidth so that to maximize their chances of being selected by the client. Empirical analysis by Murdoch and Watson [6] confirmed the efficiency of this attack.

More precisely, Murdoch and Watson investigated the relationship between the path selection algorithm and path compromise with respect to the attack's cost for the adversary. They identified the fraction of malicious Tor routers and the fraction of adversary-controlled bandwidth as important factors for predicting the adversary's ability to compromise paths.

In this research work we investigated a new attack scenario similar to attack in [6], however instead of relying on bandwidth of the injected relays, we play on the exit policies of the injected relays.

1.4. CONTRIBUTION OF THE STUDY

The main contribution of this thesis is as follows:

- We present detailed technical study of Tor protocol.
- A study of the weaknesses of the path selection algorithm.
- Investigation of a new attack on Tor based on the ports used by TCP based applications.
- Present experimental analysis showing the impact of the proposed attack on the anonymity of Tor.

1.5. RESEARCH METHODOLOGY

The research methodology followed in this research can be summarized in the following phases:

1.5.1. Phase 1: Understanding Tor

The project starts by a thorough study of the Tor protocol/network. The study focuses on:

- The literature papers: we carry out intensive survey on both old and most recent research papers on anonymity in general, and later narrow down to Tor protocol.
- The design documents: all Tor design documents are available from the Tor Project website including the TLC RFC specifications, etc. we study essentially, Tor main design specifications and path selection specification.
- The source code: Since Tor is open source software the source code is available, we modified the code to obtain some statistical information, recompiled and implemented.
- The local deployment of the network: the Tor network can be locally deployed which will constitute a valuable source of concrete information about the functioning of the protocol.

1.5.2. Phase 2: Attack Design, Implementation and Deployment

Our attack methodology consists of combining three techniques:

- Technique 1: Path selection manipulation

Focus on the manipulation of Tor path selection algorithm to make it selects the malicious Tor relays with more probability. This is a known and well-proven technique to attack Tor network. It is well established that if the path selection algorithm is successfully manipulated, the attacker has a significantly better probability to control one, two, or the three nodes of the Tor circuit.

- Technique 2: Port redirection

In network communication, every application type (http, ftp, telnet, etc.) has a port number associated to it and standardized by the IANA (Internet Assigned Numbers Authority). However, port redirection is a known trick that allows bypassing firewalls by tunneling traffic through another port number. In this attack, our approach is to push the client to use some unpopular port numbers.

- Technique 3: Injecting Malicious Relays

Injecting relays is a very popular technique to compromise circuit nodes. It has been used in most of reported attacks on Tor. In the context of this research, we use this technique because it does not require large networking resources. It only requires a certain number of relay information that we can moderately satisfy.

1.5.3. Phase 3: Experimental Methodology

In this attack we assume that the attacker is in control of a web server and has the ability to inject a certain number of Tor relays with the desired bandwidth and exit policy. Hence, the experimental setting is composed of a compromised web server and a certain number of compromised Tor relays. Then, we simulate a client that will repeatedly open circuits in the Tor network. The experiment consists in measuring the fraction of the circuits being broken by the attacker (i.e. the entry and exit nodes of the circuit happen to be injected by the attacker). We will observe the trend of this fraction as we play on:

- number of injected relays
- bandwidth of the injected relays
- exit policies of the injected relays

1.5.4. Phase 4: Attack Mitigation

This phase focuses on proposing mitigation to prevent this attack or at least make it very difficult to execute. More precisely for every exploited weakness that made the attack possible, suggest countermeasures.

1.6. ORGANIZATION OF THE THESIS

The rest of this report is organized as follows. Chapter 2 presents different anonymity Systems. Chapter 3 presents related work on Tor networks. Chapter 4 presents our proposed attacks based on unpopular ports, as well as details on simulation of Path Selection Algorithm. The implementation and experimental analysis are discussed in chapter 5. Finally, attack mitigation and conclusion are presented in Chapter 6.

CHAPTER 2

ANONYMITY SYSTEMS

This chapter is a survey of various anonymity protocols. We attempt to provide classification of anonymity systems and we present the state-of-the-art anonymity services or free software.

2.1. DINING CRYPTOGRAPHERS NETWORK (DC-NET)

The Anonymity idea was first proposed by David Chaum in the journal of cryptology 1988 where he illustrated the non-routing based anonymity communication systems “The Dining Cryptographers problem - unconditional sender and recipient unlinkability [7, 8]. Figure 2.1 below illustrates the DC-Net.

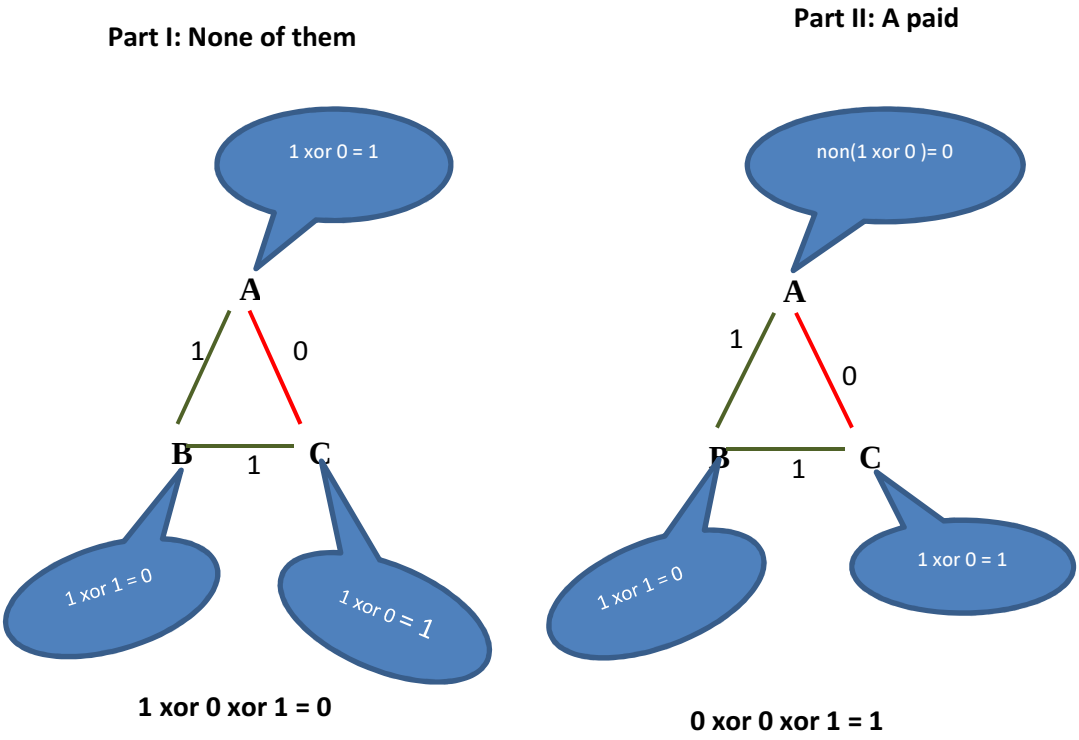


FIGURE 2.1: Illustration of dining cryptographers' Network (Dc-Net). A paid but its identity remains anonymous to B and C

The above illustration involves three cryptographers who are served food in a restaurant and were expected to settle the bill anonymously in a way that if one among the three pays the bill anonymously the remaining two cannot link the payment to the payer. However, another party not among them (says NSA) may pay the bill and in that case no anonymity is involved. So the problem is to find who actually paid the bill among the cryptographers through the use of two stage protocol. In the first stage a secret key is shared pairwise among the three cryptographers say by a toss of a coin on condition that they all follow the ethics of not revealing to each other the secret shared by each other pair. In the second stage of the protocol each of them privately XOR the two secrets shared with others and announce the result of XORing only if one is not the payer of the meal or announces the opposite of the result of XORing, if and only if one paid the bill. The third stage of the protocol involves finding out who paid the bill for the meal. If the result of XORing the three announce results by the three cryptographers is zero, then the third party not among the three cryptographers (says NSA) paid the bill and in that case no anonymity is needed, otherwise one of them is the payer but it turns out to be not linkable to any participants. In the above illustration, in the second part ‘A’ is the payer, but ‘B’ cannot link the payment to ‘A’ because, ‘B’ doesn’t know the secret between ‘A’ and ‘C’. Likewise ‘C’ can neither link the payment to ‘A’ because, ‘C’ doesn’t know the secret between ‘A’ and ‘B’. However, the protocol is extendable to n group of participants in a ring topology.

Wright et al in [16] presented a generic attack based on probabilistic approach on group of anonymity protocol models DC-Net inclusive and concluded that it lacked scalability, it is vulnerable to Denial of Service attack and required extremely low cost to attack the ring-based version of DC-net.

2.2. MIX SYSTEM ANONYMITY PROTOCOL

Prior to DC-Net mix system anonymity protocol was introduced by Chaum. This was in the late 1981 when RSA public key encryption was relatively new [7]. The main idea is that messages to be anonymous are relayed through one or more nodes called mix which originally uses RSA public key encryption. Messages are divided into blocks and encrypted using RSA. Conceptually the first few blocks are the header of the message and contain the address of the next mix node. More precisely, a mix first generates a public and private key pair and makes the public component known to clients who wish to relay messages through the mix. We let C = Ex (M) denote encryption of a message M with mix x's public key and also let M = Dx (C), denote decryption of a ciphertext C with mix x's corresponding private key.

The concept of mix was invented to prevent traffic analysis associated with mailing system. Mix system receives several mails from different client mailing servers and other mix nodes it then permutes the collected mails and sends out to mail server or another

mix in such a way that the sources and the intended destinations cannot be linked. Through the use of mix system between the mailing servers (say two mix system shown in figure 2.2), Alice prepares and encrypts the message to be sent to Bob, using Bob public key K_B appends with the Bob address, thereafter she double encrypts the resulting message with the public keys of the two mix systems as follow:

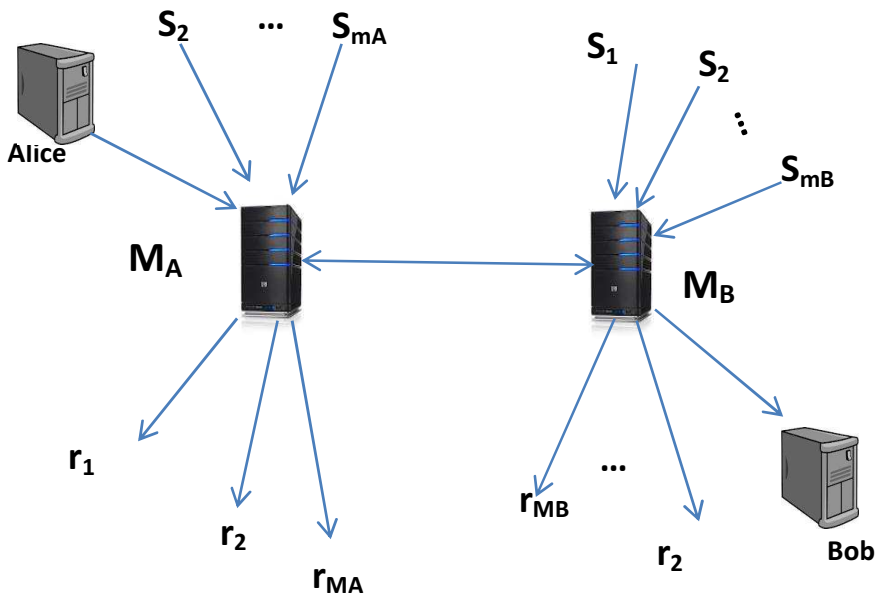


FIGURE 2.2: Mix Networks for two mixes

K_{MA} (R_{MA} (K_{MB}(R_{MB} (K_B(R_B , M) | $B_{Address}$)) | $MB_{Address}$) $\Rightarrow$ K_{MB}(R_{MB} (K_B(R_B , M) | $B_{Address}$)) $\Rightarrow$ K_B(R_B , M)

Where K_{MA}, K_{MB} and K_B are the public keys of the first mix, second mix and Bob respectively, and $R_{MA,}$ R_{MB} , and R_B are random strings of the first mix, second mix and Bob respectively added to prevent vulnerability of attacks due to identical message under the same asymmetric key, these are discarded. And $\Rightarrow$ represent transformation of the message as it passes through the Mix System. Without loss of generality, we assume that Alice sends K_{MA} (R_{MA} (K_{MB}(R_{MB} (K_B(R_B , M) | $B_{Address}$)) | $MB_{Address}$) to the first mix node, the node decrypts the incoming messages using its private key, permutes the messages and sends K_{MB}(R_{MB} (K_B(R_B , M) | $B_{Address}$)) to the second mix node, on reaching the next mix node, it decrypts the message including other messages it receives from various sources with its private key, permutes the messages and sends K_B(R_B , M) to bob.

In [17], Pfitzmann and Pfitzmann shown how an active attack can break the original Chaum's mix networks, their argument was based on the direct use of the RSA for signing and encryption, rather than the mix protocol itself, RSA has a well-known attack which exploits its mathematical bases; the integer factorization. In 1995 Kocher described a new timing attack [18] based on details knowledge of processing power of

the machine in which the RSA is implemented, ability to measure the decryption time of some known ciphertext then decryption key ‘d’ can easily be deduced.

Nevertheless, following Chaum’s mix network proposal researchers have developed diverse anonymity systems for different applications examples include Crowd’s for anonymous web transaction[9], Freenet for distributed anonymous information storage and retrieval [10], Onion Router for anonymous routing [11 and 12], Tarzan for p2p networking [13], and VPN for secure shell. While some other researchers work toward enhancing mix network for instance, [1] stated that Babel and Mixmaster are implemented based on original Chaum’s mix model and both extended the original idea of mixing. Batches of messages to provide backward feed of messages in a pool as in the case of Mixmaster, while Babel delays some fraction of messages on additional round, these together with other methods aimed to prevent (n – 1) attacks, in which an attacker sends one message in order to trace an empty mix together with (n -1) recognizable messages.

2.3. CROWDS ANONYMITY PROTOCOL

Reiter and Rubin in [9] presented a different anonymity system called Crowds for web transactions. In Crowds protocol the client request is randomly sent to one of the user participating in crowd which in turn randomly forwards it to another user or the intended

server on behalf of the originator of the request. Through this protocol, web servers are unable to learn the originator of a request because all participants are equally likely to be the originator of the request and not even the collaborating member who forwards the request on behalf of another. Crowds in [1] was described as landmark in anonymity research since its security lies on the inability of an attacker to observe the links, rather assumed to control a small fraction of nodes which can be achieved using simple link encryption and padding.

2.4. TARZAN ANONYMITY PROTOCOL

Tarzan is an onion routing peer-to-peer anonymous IP network overlay [13]. Tarzan achieves its anonymity with layered encryption and multi-hop routing, very much like Chaum original mix. It uses a network address translator (NAT) to bridge between Tarzan hosts and obvious Internet hosts. A peer initiates the transport of a stream through the network, creates an encrypted tunnel to another node and requires that the node to connect the stream to another peer. By so doing, Tarzan has a restricted network topology in which each node maintains persistent connections with a small set of other nodes creating a structure called mimics. Paths for anonymous traffic are selected in a way that they will go through mimics in order to avoid links with insufficient traffic. [1] Points out that the weak spot of mimics scheme is that selection of neighboring nodes is done on the

basis of network identifier or address which is easy to spoof in real-world networks. Clayton and Denezis reported in [1] that they found an attack against original Tarzan strategy which requires each node to know a random subset of other nodes in the network.

Another widely used onion routing anonymous network is Tor. Tor is a second generation onion routing anonymity system designed to support TCP-based applications.

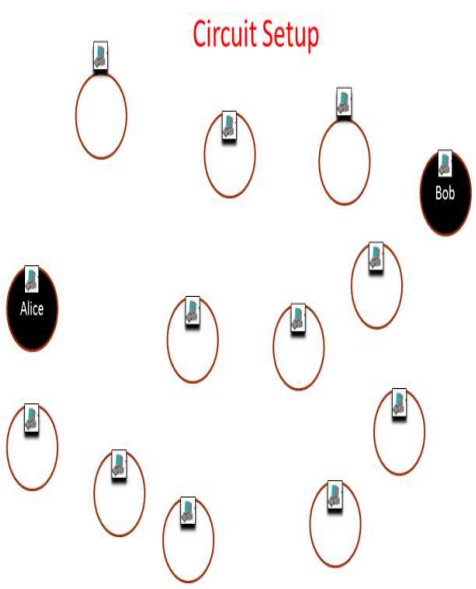
2.5. TOR PROTOCOL INNER-WORKING

Tor is the most popular anonymity network known today. It is designed with different flavors to provide anonymity to Internet users. It is open source software maintained by Tor Project. The network relies on public users to donate their bandwidth as relay. Tor relays are mainly three types: (1) entry node-first relay through which a client connects to Tor network, (2) middle node-intermediate relay which helps to extend client traffic forth and back, and (3) exit-node which submits client request to a remote server. The chosen exit node must have exit policy that supports client application port. Other Tor nodes include guard node, directory servers, autonomous servers and hidden service relays (introductory and rendezvous points).

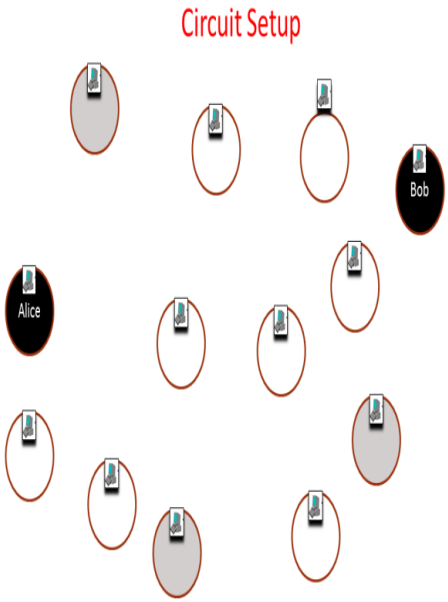
In a broader sense, for Alice (the client) to communicate with Bob (the server) anonymously over Tor networks as illustrated in figure 2.3(a), Alice's onion proxy (OP)

obtains a list of Tor relays (nodes) from directory server. From the list, it randomly selects pathway through Tor relays to Bob (see figure 2.3(b)). Starting with the selection of an exit node by ensuring that its exit policy is met, client proxy establishes session key and circuit with the first or entry node, tunneling through this circuit it incrementally extends the circuit one node at a time up to the exit node. In each step, it establishes session key with the Tor node in its pathway as shown in figure 2.3 (c, d and e).

Once the circuits are successfully established, Alice can then communicate with Bob relaying traffic through Tor nodes anonymously as in figure 2.3 (f). The OP's edge onion relay (entry node) in the circuit knows that it is communicating with Alice client and knows that it is to relay the incoming payload to the next Tor node in the path, but cannot confirm that Alice is the owner of the incoming encrypted data, neither can it say that the next node in the path is the final recipient of the data. The same applies to the next onion relay up to the exit node who knows that the message is for the Bob but cannot say who originated the communication (see illustration in figure 2.4).


Circuit Setup
Bob
Alice

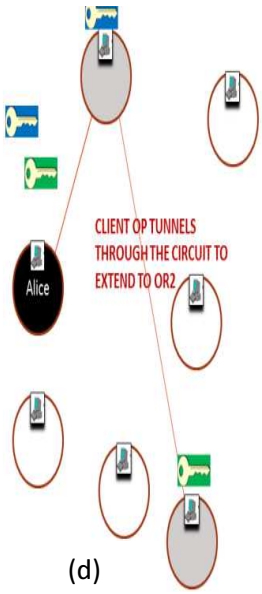
(a)


Circuit Setup
Bob
Alice

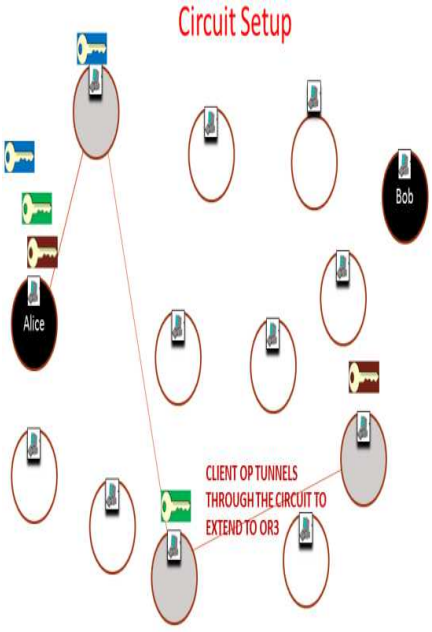
(b)


CLIENT OP ESTABLISH SESSION KEY + CIRCUIT WITH THE OR1
Alice

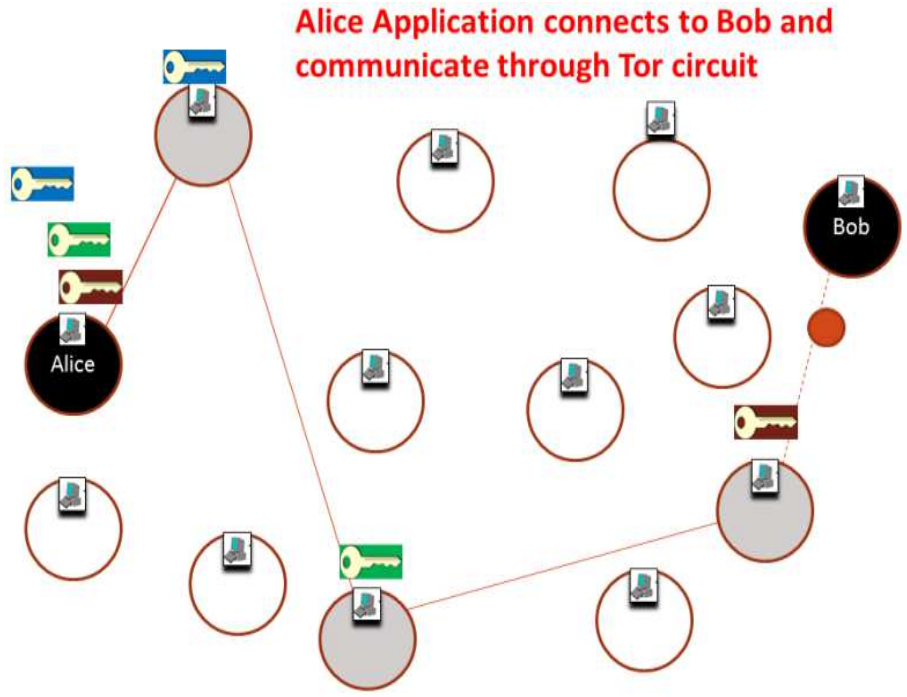
(c)


CLIENT OP TUNNELS THROUGH THE CIRCUIT TO EXTEND TO OR2
Alice

(d)

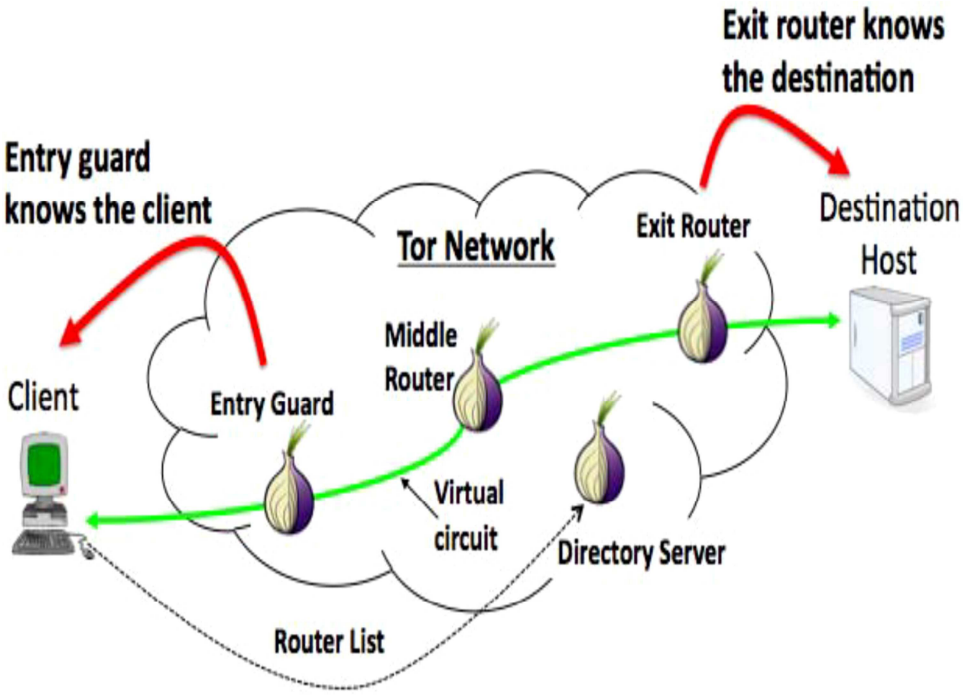

Circuit Setup
Bob
Alice
CLIENT OP TUNNELS THROUGH THE CIRCUIT TO EXTEND TO OR3

(e)



(f)

FIGURE 2.3: (a) Alice wants to communicate with Bob, (b) Alice's OP obtained list of all routers and selected path, (c) OP establishes session key and circuit with Entry onion router, (d) OP tunnels through the circuit to establish session key and extend the circuit to middle onion router, (e) OP further tunnels through the circuit to reach exit onion router, establishing session key and extends the circuit, and (f) Alice traffic passes through Tor network to Bob.



Adopted from [5]

FIGURE 2.4: Tor's system architecture and threat model

Moreover, to see the in depth description of how Tor works based on the specification obtained from [4], it is good to note that the unit of communication in Tor protocol is fixed-width "Cell" (see figure 2.5a), Cell packet mainly consists of three components:

I. The Circuit ID whose values indicate which virtual circuit the cell references to.
II. The COMMAND whose values are different commands used to communicate between client and Tor relays forward and backward.
III. Lastly, PAYLOAD which is the store for messages/data to be transmitted to another node.

Additional details of other fields and/or the different values they can accept will be cleared to us as we describe further. For the purpose of description again, we assumed that Alice (the client) wants to communicate with Bob (the server), and that Alice Onion Proxy (OP) has obtained a list of Tor relays from the trusted directory server. Also, supposed that Alice client proxy has randomly chosen only three distinctly unique relays, namely R_1, R_2, and R_3, where it chooses R_3 first, to serve as its exit OR node and R_1 as the entry OR node. The following describes circuit establishment:

a) Alice proxy generates CREATE Cell (see figure 2.5b), assigns arbitrarily unique 2- byte integer value as Circuit ID, assigns 'CREATE' as COMMAND field value, while PAYLOAD contains padding and the following; OAEP (Optimal Asymmetric Encryption Padding, which is used mainly in RSA to prevent vulnerability of short message attacks), Symmetric key K, 1st part of g^X, and 2nd part of g^X. In this step the client proxy divides the expected random number (first half of DH) which is used to create master secret in to two parts for security purpose, it uses the public key of R_1 to encrypt the 1st part of the random number, and the symmetric key K. Moreover, it uses the symmetric key K to encrypt the 2nd part of the random number and forwards the cell to R_1. The idea here is that, only R_1 could be able to decrypt the first part of the Encrypted message with her private key, and then get access to the shared symmetric key K to decrypt the 2nd part of g^X.

So, when the CREATE Cell reaches R_1, it will decrypt the first part of the message with its RSA private key and use the revealed shared key K to decrypt the second part of the message. Then combines the two part of g^X to form the complete random number (i.e. 1st half of DH) sent from the client proxy. R_1 then generates its own random number g^Y (2nd half of DH) and combines the two (g^{XY}) to form the pre-master secret (K0). Subsequently, it uses the pre-master key to generate master secret (KH) and finally, further hashing of K0 creates 100 bytes key material K (i.e. K= (KH| Df| Db| Kf| Kb)) in accordance to Tor specification schemes [4]. When this is done, R_1 sends respond to the

client by creating CREATED Cell (see figure 5c), containing the same value of Circuit ID, CREATED as COMMAND field value, and PAYLOAD contains server's random number g^Y (second part of DH), and derivative key (KH). When Client receives CREATED Cell, it uses its random number (g^X) together with the returns server's random number (g^Y) to calculate pre-master key and subsequently the master key K. It uses the agreed SHA hash algorithm with first 20 bytes of K to form the derivative key (KH) and compare with the one received in the CREATED Cell, if they are the same, then the Handshake is completed. Session key, plus circuit is established with R_1.

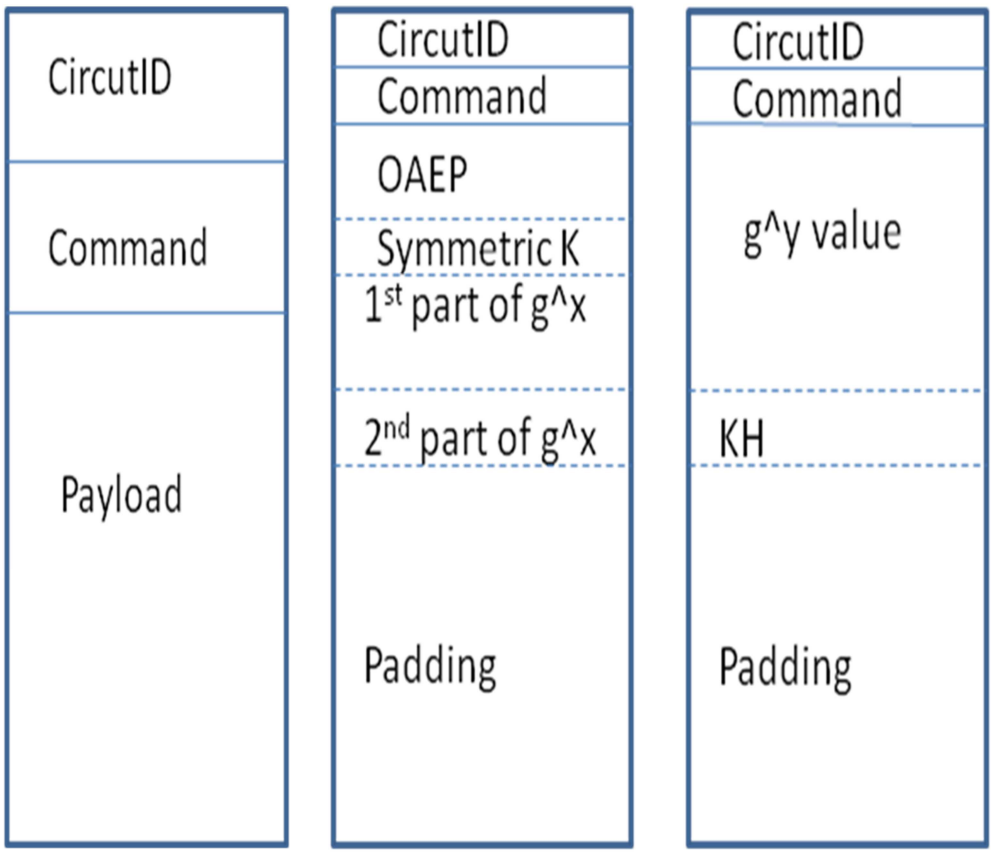


Figure 2.5: (a) A typical Tor cell, (b) CREATE Cell and (c) CREATED Cell.

CELL_LEN = 512 bytes; 2 bytes Circuit ID, 1 byte Command and 509 bytes PAYLOAD

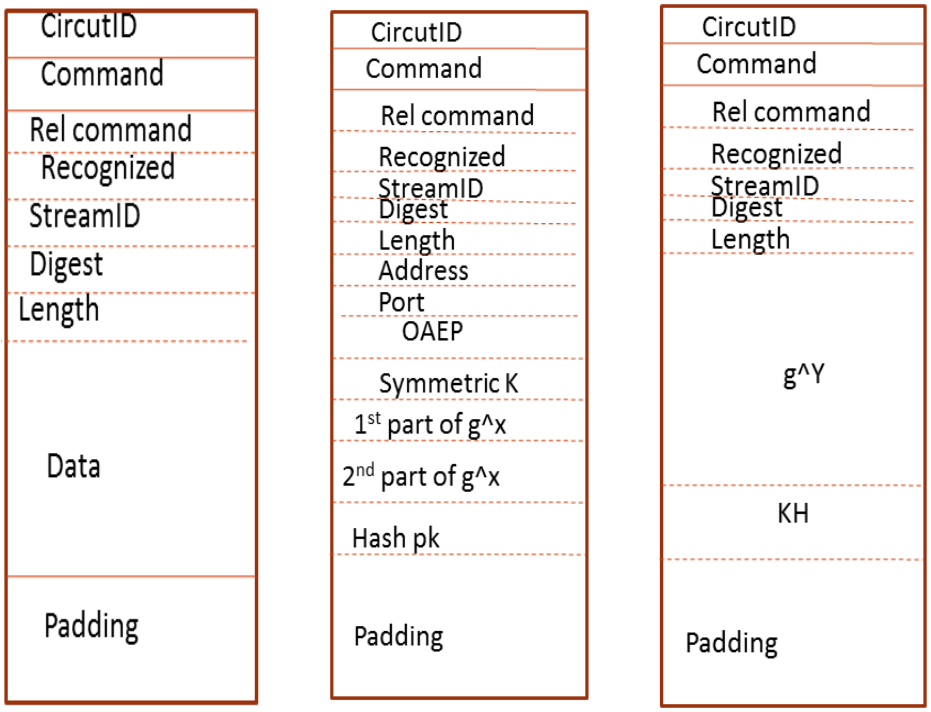


FIGURE 2.6: (a) A typical Relay Cell, (b) Relay Extend Cell, and (c) Relay Extended Cell

b) To tunnel through the circuit established with R_1, and extend to R_2, Alice client creates a RELAY Cell (see figure 2.6(a)) with RELAY as COMMAND field value, and PAYLOAD containing two messages as shown in figure 2.6(b).

The first message is unencrypted and is to be used by R_1 for further instruction about the nature of RELAY command. This unencrypted message contains RELAY EXTEND as REL-COMMAND field value, an integer digit greater than 0 (which means R_1 is to process the cell and forwards it to another Relay node) as RECOGNIZED field value, zero or an arbitrarily chosen ID by OP (assigned to a relay cell of the same circuit and used to determine cells belonging to the same data stream) as STREAM ID field value, DIGEST field value is 4 bytes of running digest seeded from Df (forward digest) shared with R_1, and the number of bytes in relay payload for real payload data as LENGTH field value. Address and port refers to ipv4 and port number for the next relay node in the path (R_2). The second message contains a CREATE information - OAEP, Symmetric key K, 1st part of g^X and 2nd part of g^X, similar to the one described in a) above. Interestingly 1st part of g^X and symmetric key k are encrypted with R_2 RSA Public key, while the 2nd part of g^X is encrypted with symmetric key k. The entire messages in the payload are then encrypt with the forward key (kf) shared with R_1.

The RELAY Cell is transmit to R_1, on receiving the cell R_1 checks the Circuit ID and determines if it has corresponding circuit along that connection (which it has in this

case), then decides if RECOGNIZED field is zero (which is not) and ensures that other conditions hold. R_1 executes RELAY EXTEND command by creating CREATE Cell, generates unique 2 bytes integer Circuit ID not yet used on the connection, and encloses the second part of message it received as payload from RELAY Cell into the CREATE cell as PAYLOAD and transmits to R_2.

In returns, R_2 decrypts the first half of DH using its private key and shared key k in a similar way described previously, it then creates CREATED cell containing its own randomly generated half of DH (2nd half of DH) and computes KH (which is the 20 bytes derivative key) as Payload data. It then sends this cell backward to R_1 as shown in figure 2.6(c). R_1 will then replaces the content of RELAY Cell PAYLOAD with RELAY EXTENDED as REL-COMMAND field value, 4 bytes digest seeded from Db (backward digest) shared with OP as DIGEST, 0 as value of RECOGNIZED field and the payload handshake data from R_2 CREATED cell as well as the new value of LENGTH field.

The PAYLOAD is encrypted using shared Kb (backward key) and the cell transmits back to OP. When Client OP receives RELAY EXTENDED cell it decrypts the payload using Kb (backward key) it shares with R_1, it then observes the Circuit ID, stream ID to ensure that there are matches, observes that the Recognized field is zero and the Digest value equals. Thereafter, it uses its half of DH with the received half of DH from R_2 and calculates full DH key (pre-master key) using the key to derive K (the master key). Then, it compares the generated derivative key KH with the one received in payload, if they are

the same, the handshake is completed, session key established and the Circuit is extended to R_2.

c) To further extend this circuit to R_3 which is the exit node in this case, Alice client creates a RELAY Cell similar to b) above. Unlike the above, the PAYLOAD is firstly encrypted with the Kf (forward key) shared with R_2 forming the inner onion layer, and then with forward key shared with R_1 forming the outer onion layer. The relay cell is sent to R_1. On reception of the Relay cell, R_1 decrypts the outer onion layer with the forward key shared with OP and observes the content of PAYLOAD, it further processes the data in the PAYLOAD if it recognizes it. Otherwise it forwards the Cell along the circuit. R_2 receives the RELAY cell, observes the Circuit ID, decrypts the inner onion layer, and uses the unencrypted message in the PAYLOAD to process the RELAY Cell. It observes the Stream ID and Rel-Command field values. With the value of RECOGNIZED field not equal to zero and observes that other condition been hold, R_2 creates CREATE Cell with a unique Circuit ID, and encloses the encrypted data of RELAY EXTEND cell payload into the CREATE Cell PAYLOAD and sends to R_3 after observing the port number and validity of the address. R_3 receives the CREATE cell, use its RSA private key to decrypt the first part of the data in the PAYLOAD, which is the same with the one described several time above - OAEP,

symmetric k shared key, 1st part of g^ X and use the revealed symmetric key to decrypt the second part of the data in the PAYLOAD, which is meanly the 2nd part of g^X.

R_3 then creates CREATED Cell contains its own randomly generated half of DH (2nd half of DH) and computes KH (which is the 20 bytes derivative key) as Payload data and sends backward to R_2. In sequence R_2 retrieves the CREATED Cell Payload, encloses in RELAY cell, replaces the command with RELAY EXTENDED encrypts the entire payload with backward key (Kb) shared with OP, and sends backward to R_1. R_1 further sends backward the RELAY cell once encrypted with its backward key shared with OP. On receiving the Relay cell, Alice client decrypts the outer layer with the backward key shared with R_1, and decrypts the inner layer with the backward key shared with R_2 to reveal the RELAY EXTENDED Cell. It observes the Circuit ID, stream ID to ensure that there are matches, observes that the Recognized field is zero and the Digest value equals. Thereafter, it uses its half of DH with the received half of DH from R_3 and calculates full DH key (pre-master key), using the key to derive K (the master key). It then compares the generated derivative KH with the one received in payload. If they are the same, the handshake is completed, session key established and the Circuit is extended to R_3. And this completes the session keys and circuit establishment. What follows is the data exchange via end-to-end TCP connection with Bob (the server).

2.6. TAXONOMY OF ANONYMITY SYSTEMS

Anonymity systems can be divided into different types either based on anonymity objects or based on mechanisms of operation. Based on the anonymity object, the anonymity systems are divided into three classes namely:

- Sender anonymity which conceals the relationship between message and its sender
- Recipient anonymity which conceals the relationship between message and its recipient
- Relationship anonymity which conceals relationship between sender and the recipient.

However based on the mechanism of operation anonymity can be divided into two types:

- A non-routing-based anonymity communication system like Dining Cryptograph technology (DC-Net) which ensures the unlinkability of sender anonymity, recipient anonymity and relationship anonymity.
- The other type is the routing-based anonymity in which data are passed through one or more transmitting nodes between the sender and the recipient, while the nodes can rewrite, fill and transmit data packets to hide the source of data packets and their relationship between input and output. Examples of routing-based anonymity are mix system, Onion Routing, Tarzan and Crowd, etc.

- Moreover, these routing-based anonymity systems are further divided into high latency and low latency systems based on Internet applications they support. Common Internet applications such as Web browsing, video teleconferencing, file transfer protocol (ftp), remote login (Telnet), emails and broadcast all of which use IP (Internet protocol) as the transmission mechanisms. These have different performance indices because of their requirement on network bandwidth, responsiveness, tolerance to communication noise and implementation techniques.

  Thus, mix systems which are suitable for low responsiveness application such as email, are referred to as high latency systems, while Onion routing systems which are suitable for high responsiveness applications such as web browsing, chatting, ftp, etc. are referred to as low latency system.

2.7. FREE ANONYMITY SOFTWARE

While they may not provide 100% anonymity online, these free anonymity software can be used in places where there are high need for privacy. Places like Internet café, libraries, Airport, schools, workplaces, public and prepaid Wi-Fi hotspot. Pending on individual needs they are good for home usage in order to avoid Internet eavesdroppers of all kinds.

2.7.1. Cross-Platform Solutions

- JonDo originally known as JAP is a VPN client written in java and routes data across the JonDo networks. JAP was originally developed as part of project of the Dresden University of Technology, the University of Regensburg and Privacy Commissioner of Schleswig-Holstein, Germany [14]. It sends requests through a cascade and mixes data streams from different users in order to further conceal data to outsiders. Unlike Tor, the mix cascades (set of anonymization proxies) used by JAP are known individual organizations and users themselves may choose among the operators they will trust. JAP name was changed to JonDo in 2007 after financial backing of the original project ran out. Now JonDo is maintained by members of the original project team and it provide free services as well as enhance commercial services.
- Vidalia is a Tor client that bundle Tor software and Polipo and routes traffic across Tor network. Vidalia provides Tor users with options to donate their bandwidth as Tor relay, it constantly change user IP to disguise user Internet trail, and provide better anonymity.
- PocketiX.NET is an academic, non-profit online environment for PocketiX VPN offered by SoftEther Corp. Unlike Hotspot Shield and other VPN clients, it is not based on OpenVPN, rather uses a proprietary system. All traffic over PocketiX.NET is encrypted using SSL, just like other VPN. Since it is run by

university of Tsukuba in Japan the actual speed user get depends on geographical location in relation to Japan.

- JanusVM is a VMware based Tor/Privoxy/Squid/OpenVPN client that allows user to browse Internet without censorship. It combined the power of Tor, Privoxy, Squid, and openVPN to increase anonymity. JanusVM require VMware Player to work, it has advanced filtering capabilities for modifying web page content, managing cookies, controlling access, and removing ads, banners, pop-ups and other intolerable Internet junk.
- ProxPN is a lightweight VPN client based on openVPN, currently have servers located in USA, thereby making speed varies pending on your location. To use the services, users are required to register first and traffics are tunneled through encrypted connection through VPN servers, then out of the Internet.
- USAIP is VPN based on PPTP and L2TP protocols, though not bound to any operating System, Microsoft played major role in its development and it can be used on Mac OS and Linux as well. The program effectively tunnels all traffics through this service, thereby making all of your internet activities anonymous.
- Your Freedom is java based tunneling solution. It has plain and simple looking java GUI with servers located worldwide, support UDP and allows you to play games online using the tunnel.

2.7.2. Window-based Solutions

- xB Browser also known as XeroBank Browser, previously known as TorPark is a Firefox and Tor bundle it is often refers to as all-in-one solution, it has a vary speeds as it is the case with all Tor connections, however disabling images will speed up things. It is a window based solution.
- Hotspot Shield, also known as AnchorFree is a free VPN based on openVPN. It encrypts all Internet activities of user and not just a browser as is the case for JonDo and Tor. It ideal for users requesting anonymity for other applications without the use of proxy. Because it doesn't collect any identifiable data about user, it is the world must trusted VPN used by over 10 million people [23]. Users are rests assure that their privacy is guaranteed. Another good feature of AnchorFree is high speed comparable with commercial solution on the normal ground, but often display advert on a requested page when using browsers such as IE, Chrome, Safari and Opera, etc. But it is completely free.
- AdvTor is developed with Tor advanced user in mind, a powerful alternative to Vidalia bundle. It allows pretty well customizing of Tor connection from adjusting bandwidth to manual entry / exit node selection. One of its best features is the ability to force a program such as instant messaging client to use Tor connection. Good to note that Tor is TCP based anonymity protocol which

doesn’t support UDP protocol. So any application that uses UDP protocol will be sent unencrypted over user’s network connection without the feel of Tor.

- SecurityKiss is a simple window VPN based on openVPN, it has relatively fast speed depending on location. SecurityKiss persistently say they do not keep personally identifiable data about their users, but only log of user’s IP addresses, connection/disconnection times and traffic volume. They used the 128-bit blowfish algorithm to encrypt session data and use 1024-bit RSA certificates for session keys, so as to make user be rests assured that no eavesdropper can steal user’s data.
- UltraSurf is a VPN program for Windows with server located in USA. The program collect user IP, the property of browser and / or your computer, the number of links you click within a site, state or country from which you accessed the site, date and time of your visit, name of user’s ISP, web page you linked to our site from, as well as page you viewed on the site. Some antivirus program false positively identifies UltraSurf as virus / Trojan since it can pierces through firewalls.
- CyberGhost is a VPN client with server in Germany. They used 128-bit AES encryption on all connections to ensure a high level of anonymity and use special data compression techniques to make speeds faster.

- FreeGate is a free proxy offered by Dynamic Internet Technology Inc. The program offers unrestricted access for Chinese users, but restricted access to other users.

2.7.3. Linux-based Solutions

- Incognito Live System (Amnestic) is a Debian-based Live CD. Vidalia is bundled with the CD which forces all connection to go through Tor network. Absolutely no data is stored when using the Live CD unless explicitly configured to do so.
- Estrella Roja (Red star) is a Linux based Live CD which comes with Tor and Privoxy. All data are route through Tor network, this distribution is in Spanish.
- Privatix is by design almost identical to Incognito Live System, significant different is that Firefox with Tor button is the default browser.

For the taxonomy of the above free anonymity software see appendix. Details information such as speed rating, anonymity rating, usage allowance, logging level, server location, supporting OS, Web of Trust (WOT) rating, and RAM usage are provided in the taxonomy.

CHAPTER 3

SURVEY OF ATTACKS ON Tor

Since its deployment in the late 2003 several efforts have been made to analyze the anonymity claim of Tor through different attacks on Tor protocol, some of which resulted in patching and updating of the program. Several attacks claim have been presented in the literature. The need to have a standard means of measuring the efficiency of such attack claims is highly needed.

3.1 TRAFFIC ANALYSIS

Among the earliest attacks on Tor protocol was Traffic Analysis. The attackers demonstrate the possibility of obtaining information from Tor without even tempering with the core relays. This attack utilizes the tradeoff between the need for low latency and complete anonymity.

In [19] Murdoch and Danezis presented Low-cost traffic analysis that can be mounted on Tor by looking at a different behavior of Tor relays. Since in Tor cells from different streams can be sent out in round robin fashion which shows that higher load increases latency of all other connections. So by routing a connection through specific Tor nodes and measuring the latency of messages, an attacker can get estimate of the Traffic load on the Tor node and thereby possibly trace the path way. Moreover the possibility of linking transactions based on this attack was expressed, but the result of the experiment shows that they were only able to reveal the pathway. This work was done in 2004 when Tor relays were relatively fewer in numbers compared to the present number of Tor relays. We think it would be very difficult to realize the same result today, even without patching up Tor.

In another work [20] Murdoch and Zieli´nski demonstrated how traffic analysis can be used on Internet exchange points by developing traffic analysis techniques which work

on sample data collected from Internet exchanges (IXies) and use Bayesian methods to obtain the best possible inference. This work revealed the vulnerability of Tor on a specific traffic while guaranteed safeness on others.

Prateek et al in [24] presented a traffic analysis of low latency system through the use of throughput fingerprinting. The attack is based on observing throughput of a Tor circuit over multiple connections, and is able to say if two concurrent TCP connections belong to the same Tor user. The attack utilizes information leakage through circuit throughput. Using fingerprint the author determines whether two circuit share a common sub-path and finally analyzed their correlation using simple statistical test. The probabilistic observation was combined over multiple circuits, and leads to de-anonymize guard relays.

3.2 ATTACKS ON Tor HIDDEN SERVICES

Tor Hidden service was first deployed in 2004, it mainly aimed at providing protection to dissident and activists in a danger zones and was recommended by both Electronic Frontier Foundation and Reporters without borders.

However in its earlier time Øverlier and Syverson [21] presented an attack on how to locate a hidden server by demonstrating how a single malicious node can reveal the hidden server, this attack utilizes packet counting, incoming/outgoing packets matching,

and etc. But the report seems to be presented at belated time after the introduction of guard node which protects Tor against such attack.

In another work [22] Murdoch presented a possible attack that can reveal the Hidden server through their clock skew. The basic idea is to utilize temperature difference between the CPU of working nodes and that of idle nodes which has effect on clock skew, clock skew can be observed remotely by attacker to identify active and inactive nodes using timestamp and sequentially give clue about the likely region the hidden serves are located by exploiting time zone difference.

A technique to prove that a confiscated hidden server has hosted a particular content was presented in [25]. The technique relies on leaving a timing channel fingerprint in the confiscated server's log file. It is based on sending HTTP requests to the hidden machine and storing "date field" of the responses obtained. So to detect the fingerprint a logged entry is look for in a window around the time that appears in these responses. While, there are false positive results, the author provide analytical expression for the probability of detection and the false positive. One important issue that the paper did not address is how to reveal the Tor hidden server, as far as we know, no previous research on how to reveal the hidden machine has proved viable. Most prominent of the hidden server attacks was that of [21] which is not applicable to the present Tor network and that of [22] which only give idea on the region the hidden machine is located, but not revealing the machine. As we know the hidden server has no customary IP addresses, it mainly provide services as a local host through Tor network, which is very hard to detect.

3.3. CELL-BASED ATTACKS ON Tor

The unit of communication in Tor network is Cell. A Cell has a fixed width with length of 512 bytes, exactly the size of typical TCP packet. Because of the fact that different types of Tor cells are used to carry different type of information, researchers exploit vulnerability due to the nature of information they are designed to transmit.

Ling et al in [26] presented a cell counting based attack on Tor. This attack was inspired by extensive experiment that reveals that the size of IP packets in the Tor network may not be fixed. The attack has potential of revealing anonymous communication relationship among clients, a malicious exit router randomly intercept a traffic from server to the client insert a unique signal that can change the cell counts and channel the cell to client along the circuit, however, malicious entry node detect the difference in the received cell and the enclosed signals. The core target of the malicious exit router is to intercept a CELL_RELAY_DATA which is the type of cell used in Tor network for transmitting data stream. Furthermore, the authors presented a situation where a malicious exit router only can be used to successfully lunch the attack based on sniffing the packets transmitted between client and entry node - but we feel this seems to be unlikely, because exit node is unaware of the entry node, so even if attacker wants to sniff

packet between entry node and client, he may ended up with a wrong packet since is hard to pinpoint the real “client – entry” peer of interest.

Moreover, a New Replay Attack against Anonymous Communication Networks was presented in [27]. The replay attack is based on replication of Cell which is the unit of communication in Tor. The attack involves the use of malicious entry node which duplicates cells from the client and traverses both cells across the middle node to the exit node, present of duplicate cells at exit node result to recognition error due to disruption of normal counter. The attack is successful if attacker’s accomplice from the entry node controlling the exit node detect such decryption errors, thereby exposes the communication relationship between client and server. This attack was implemented on Tor and validated. Result of the experiment indicated the feasibility and effectiveness of the attack.

3.4. PERFORMANCE ANALYSIS

The research direction in Tor nowadays tilted towards enhancing the status quo through network performance analysis while maintaining complete privacy. In this direction, Murdoch and Watson [6] presented a metric for security and performance in low-latency anonymity systems and compared different path selection algorithms and analyze the algorithms using two metrics, vulnerability of being compromised with respect to

attackers resources claim and expected latency. Outcomes of the analysis indicated that Tor default path selection based on high bandwidth claims improve performance as well as anonymity compared to the supposedly more secure Tor uniform path selection algorithm.

A case Study on Measuring Statistical Data in the Tor Anonymity Network was presented by Loesing et al [29]. In the paper the authors provided guidelines for safety measurement of Statistical data in the Tor Anonymity Network. However, the paper addresses two issues concerning measuring two types of sensitive data in the Tor network for performance evaluation; measuring client IP addresses and exiting port. The guidelines include:

- Data Minimalism-minimizing the amount of statistical data required for solving a particular problem.
- Source Aggregation-sensitive data should have a short time span, aggregation should be the source of data and always approximate exact event counts.
- Transparency-public discussion of all algorithms for data gathering is essential before deployment and all measured statistical data should be publically available to avoid gathering of too sensitive data.

[28] Presents a new method for path selection which allowed performance-improved onion routing. The authors believe that this new proposed method can be used to increase performance and security of the Tor network's clients and equally serves as input for

improved designs of future anonymity protocols. The metric is based on actively measured round trip time and estimations of available link-wise capacities through passive observation of throughput.

In [30] Norman et al proposed an algorithm that detects DoS in Tor network, using graph theorem to model the Tor network and mathematical theorem they were able to prove that the algorithm is capable of detecting DoS in Tor network.

A new anti-misbehavior system for Tor network was proposed in [31]. After critically evaluating the existing Tor exit policies which the author claimed that it is insufficient to mitigate misused of Tor services. The new Anti-misbehavior system will contain two blacklists-global and local blacklists and three protocols-reporting misbehavior, building blacklists and blocking misbehavior user protocols. Global blacklist is built according to every Tor relay local blacklist and maintained by trusted directory servers. On regular basis, every Tor node reports local blacklist to directory server and the directory server update the global blacklists and distribute the global blacklists to every Tor node. The authors provided a comparative evaluation of the new system by comparing the Tor exit policies against the proposed new anti-misbehavior system using three metrics - user experience, performance and anonymity. Though no experiment was carryout, one important issue regarding the applicability of the proposed system that was missing in the paper is the case of two important error measurements. We observed that the case of false negative in which a genuine user is blacklisted and a false positive in which wrong user is undetected was not giving account anywhere in the paper.

Optimal Path Length for Tor was studied in [34]. The authors investigate path length design decision and evaluate both two–hop and three–hop paths through experiment. The evaluation was based on two metrics. Security – by simulating Tor default path selection for two-hop and three-hop paths, result shown that two-hop paths are less vulnerable to circuit compromise compared to three-hop paths, but they are more vulnerable to adaptive surveillance. However, in terms of performance – the authors simulate real clients making http request for some popular websites and measure their downloads time, performance result shown that the download time for three-hop path is almost twice as the expected download time for two-hop path. Despite the outcomes of the evaluation the authors concluded that the argument is not strong enough to reduce the length of Tor path to two-hop.

3.5. ATTACKS ON TOR PATH SELECTION ALGORITHM

Low resource routing attacks against Tor was presented by Bauer et al in [2], in their work they demonstrated how much extend is the routing selection optimization for performance exposed Tor protocol to end-to-end traffic analysis attack from non-global adversaries with minimum resources. Their approach involves compromising guard and exit nodes, by injecting few nodes with high bandwidth and high uptime claims, then

using their end-to-end traffic analysis method to associate the client's request to its destination before any payload data is sent.

Browser based attacks on Tor was presented by Abbott et' al [24].The main idea is that the attacker can trick a user's web browser into sending a peculiar signal over the Tor network which subsequently can help identify the user's identity using traffic analysis. In the paper they described how a malicious node acting as exit node, when selected by a client can insert a JavaScript code into unencrypted payload which is run on client machine and can generate identifiable signal pattern that can be detected by the server.

Predicting Tor path compromise by exit port was presented by Bauer et al in [5], the core idea is to exploit the role of ports in compromising routing path based on the type of traffic being propagated. They achieved this by injecting exit nodes with different exit policy which makes client to select an exit node with high bandwidth and satisfy its exit port requirement. Through these they were able to analyze different applications. The research project we propose deviates from their work since our attack model goes beyond the passive attacker, here we considered a more active attack that tries to force the client redirect its traffic through unpopular ports.

A practical Congestion Attack on Tor Using Long Path was investigated in [32], the attacker relies on using malicious exit router exiting HTTP traffic to inject JavaScript to the client which request user browser to perform http request every second and constructing a long circuit that include a particular relay repeatedly. This can be seen as

constraint of Tor path design which allows construction of circuit of arbitrarily length. Subsequently, using statistical evaluation of difference in latencies the attacker will determine with high probability the relays that make up the circuit.

[33] presents a potential HTTP-based application level attack against Tor which takes advantage of Tor design feature and vulnerability of HTTP to man-in-the-middle-attacks. It employed a malicious exit router to act as man-in-the-middle-attacks to client's communications. A respond usually a webpage to a request made by the client through malicious exit router can be modified or inject a new forged webpage which makes the client's web browser to initiate malicious connections to get those hidden objects, subsequently, malicious entry router will detect such distinctive signal. Through traffic analysis an external attacker can be able to correlate the traffic between the client and the webserver.

Manils et al in [35] presented attacks that target P2P protocol, particularly accessing BitTorrent with Tor. The attack explained how the identity of user can be reveal due to information leakage. The authors presented three techniques on how malicious exit router can de-anonymize BitTorrent traffic. Two of the techniques are completely passive attack that relies on information leakage of the application itself, while the third technique is an active attack that exploits the lack of authentication in the BitTorrent protocol.

CHAPTER 4

PRESENTATION OF THE PROPOSED ATTACK BASED ON UNPOPULAR PORTS

In this chapter we present a new attack scenario that focuses on Tor unpopular ports and combines inspirations from both Abbott et al [5] and Bauer et al [6]. Our work is similar to [6] in the sense that we investigate the relationship between the application layer protocol and the resources needed by the adversary to compromise path. The difference lies in the definition of attack scenario. Their attack scenario is passive. It involves injecting malicious entry and exit routers with high bandwidth claimed. Our attack however, is more active as it intends to reveal the identity of a Tor client connected to a compromised webserver. The attacker takes advantage of unpopular ports to play on the Tor path selection algorithm to select from few set of relays in which the attacker controlled a significant fraction. Our attack differs also from [5] since their attack is

browser based and tries to trick the user's web browser to send peculiar signals over Tor network. Our attack on the other hand, involves pushing a Tor client to open a new connection through an unpopular port.

4.1.ATTACK SCENARIO

In this attack scenario (depicted in figure 4.1) we assume that a client uses Tor network to connect to a compromised server. This means that the server provides a certain web service and a visitor uses Tor to hide his real routing identity, so that you hardly know who visited the site and from where. However, the compromised server injected a program into a requested page that will force the client to open another communication through one of the Tor unpopular ports. This way the chances that the client will choose one of the adversary injected malicious exit nodes with high self-advertised bandwidth and exit policy that accept only the requested Tor unpopular port increases.

The description of the attack is as follows:

I. The attacker controls a compromised web server which has the ability to inject a program to any client visiting the site.
II. The attacker injects some large number of malicious exit routers that accept a particular unpopular port, say port 1214, with high bandwidth claims.

III. The attacker injects some large number of malicious entry routers with high bandwidth claims.

IV. A client connects to the compromised web server anonymously using Tor network.

V. The server injects a hidden JavaScript into the requested web page by the client.

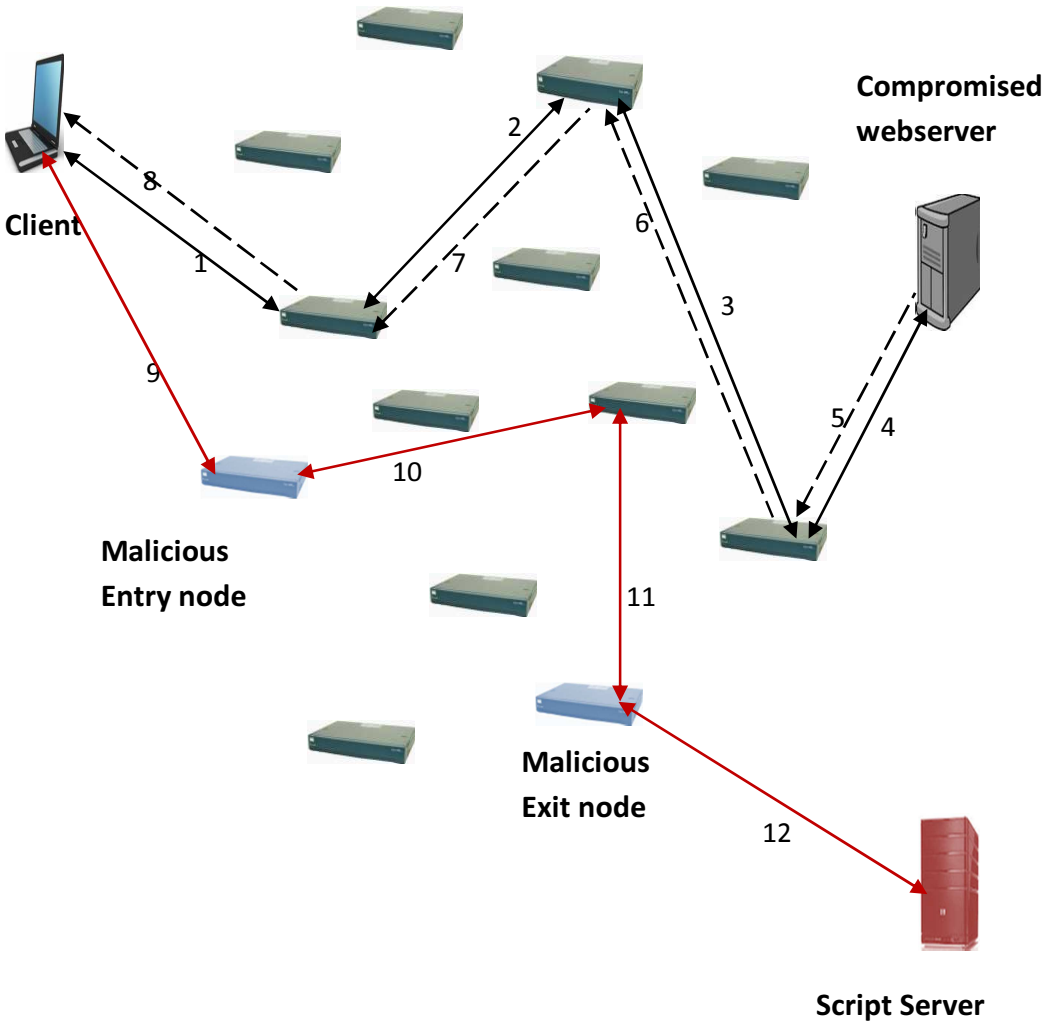


FIGURE 4.1: Our Port Redirection Attack Model; first the Web Server is compromised by storing a Script. A client uses Tor via path (1-2-3-4) to make HTTP request; the response from compromised server contains an embedded stored Script (5-6-7-8). On the Client side the Script opens a new connection torrified through Tor network sending request (9-10-11-12) via unpopular port listen by Script Server.

VI. On reaching the client this hidden Script opens a new connection through Tor network sending repeated request through port 1214.

VII. The attack is successful when the client passes traffic through both malicious entry and exit routers.

To make this attack successful we need to answer some pending questions such as how could the attacker compromises the webserver? If the attacker succeeded in compromising the webserver and injects the script into the client machine how can the script forces the client to open a new connection through Tor network to a remote server using unpopular port?

4.1.1. Method for Compromising a Webserver

Here we considered two possible ways an attacker can compromise the webserver pending on the type of attacker in question. The internal attacker can simply find a way to modify a particular webpage in the webserver with appropriate authorization, so that a malicious script is stored. Another possibility involves external attacker to discover vulnerability due to poor programming practice and use cross-site scripting like (XSS-like) technique to inject a script into the webpage. This script is then stores in the page

unknown to the server. Subsequently, each time a user visit the site the script runs in his machine.

4.1.2. How to force a client to open a new connection through Tor

Since the intention of this attack is to force a client browser to open a new connection to a remote server using an unpopular port. The attacker may decide to use a WebSocket technology which is a web technology that runs bi-directional communication channels over a TCP socket enough to tunnel Tor traffic [39]. This technology has been standardized by both IETF and RFC6455 [37]. One important feature of WebSocket protocol is the ability of its client side implementation to sense if user's web browser is configured to use proxy to connect to a remote server and port and uses HTTP CONNECT to setup a persistent tunnel. This property of WebSocket protocol is essential to the core idea of our attack. WebSocket API has been standardized by W3C. Socket.IO (is a WebSocket API) provides a method that can push traffic from client to server in an efficient manner using simple syntax [38]. Apart from that, WebSocket technology is accessible to JavaScript in a newer version of web browsers such as Firefox, Chrome, etc.

A simple client side JavaScript implemented to repeatedly send requests to a remote server listening to one of the unpopular ports, when embedded into HTTP response from the compromised webserver, will force the client onion proxy to open a new circuit. Because, it is unlikely that the Tor exit relay used by the client to connect to the

compromised webserver will relay traffics that warrants the use of any unpopular port. This way Tor path selection algorithm is left with an option of selecting from small set of exit relays that are ready to accept the unpopular port.

```
// create a socket instance
var socket = new WebSocket('ws://localhost:6969');
// Open the socket
socket.onopen = function(event) {
        // Send an initial message
        socket.send('I am the client and I\'m listening!');
        // Listen for messages
        socket.onmessage = function(event) {
                console.log('Client received a message',event);
        };
        // Listen for socket closes
        socket.onclose = function(event) {
                console.log('Client notified socket has closed',event);
        };
        // To close the socket....
        //socket.close()
        };
```

FIGURE 4.2: Client side WebSocket API Snippet

4.2. Tor PATH SELECTION SIMULATION

In order to investigate the relationship of Tor path selection algorithm and the amount of resources needed to make this attack successful, we developed a simulation that adhered to default Tor path selection specification as provided by the Tor Project. There are two parts of Tor path selection algorithm as elaborated in [2]. The Entrance Router Selection Algorithm which was incorporated into path selection algorithm with the introduction of Entry Guard, in which a client automatically chooses a set of onion routers flagged as 'fast' and 'stable' by the trusted directory servers. And the second part of Tor path selection algorithm is the Non-Entrance Router Selection Algorithm used for selecting subsequent routers in the circuit. In this work we simulated the Non-entrance Path Selection Algorithm since it is optimized to favor router selection with high bandwidth and high uptime.

4.2.1. Non-Entrance Router Selection Algorithm

This algorithm was optimized to favor router with high bandwidth for network performance reasons (see the Pseudo-code in figure 4.3). The algorithm has all known onion routers, *router_list* as an Input, and produces a randomly chosen router, weighted

toward the router advertising the highest bandwidth. At the beginning the algorithm compute total bandwidth (B) for all the available routers in the *router_list*, subsequently choose pseudo random number C between 1 and B. For each onion router from the list, a router is selected and its bandwidth is added to a variable T, if variable T is greater than C then the onion router is chosen for inclusion into the path provided the Tor path selection constraints are met. In the other hand, if T is less than C then more onion routers are selected and their bandwidth added to T until T is greater than C. From this we can infer that the more bandwidth an onion router self-advertised the greater the probability that the router is to be chosen, since the algorithm assign weight to onion router based on probability distribution that tilted towards the magnitude of router's self-advertised bandwidth.

Input: A list of all known onion routers, *router _ list* ← 0

Output: A pseudo-randomly chosen router, weighted toward the routers with highest self-advertising bandwidth

$B \leftarrow 0$
$T \leftarrow 0$
$C \leftarrow 0$
$i \leftarrow 0$
router _ bandwidth ← 0
bandwidth _ list ← θ
For each *router* $r \in$ *router _ list* **do**
    router _ bandwidth ← *get _ router _ advertised _ bandwidth* (r)
    $B \leftarrow B +$ *router _ bandwidth*
bandwidth _ list ← *bandwidth _ list* ∪ *router _ bandwidth*

end

$C \leftarrow$ *random* _ int$(1, B)$

While $T < C$ **do**

    $T \leftarrow T +$ *bandwidth _ list* $[i]$

    $i \leftarrow i + 1$

end

return *router _ list* $[i]$

FIGURE 4.3: Non-Entrance Router Selection Algorithm

Details path selection specification can be seen in [4] however, router selection algorithm chooses router with the following constraints

I. All routers in a path must be unique- no router is selected twice for the same path.
II. All routers in a path are chosen from different family, no any router is of the same family with another router in the same path.
III. By default, only one router is chosen from a given /16 subnet.
IV. Routers chosen for a path must be all running and all valid, except otherwise configured by default.
V. The first router on the circuit must be flagged as entry guard by a directory server
VI. The exit router selected must support connection to client's chosen destination host and port.

In all cases, the choices for entry and exit routers are based on considerably large bandwidth. Too much bandwidth above one-third of the total bandwidth of all routers in the network may lead to rejection of an onion router, while router with too low bandwidth may not be favored by router selection policy.

In developing the simulation we make the following assumptions:

- The use of Entry Guard is disabled.
- We assumed that all routers to be used in the simulation are valid and stable
- We assumed that all routers to be used are from different family
- All routers to be chosen are from different / 16 subnet

CHAPTER 5

EXPERIMENTAL ANALYSIS

In this chapter we describe the experimental procedures we have adopted to show the workability of our proposed novel attack idea. The first part of this chapter presents our developed prototype implementation of the attack. It contains the implementation of compromised webpage and how to push a Tor client into opening a new connection through Tor network that require the use of a particular unpopular port. While the remaining part of this chapter discuss the simulation of Tor default path selection and how an attacker with significant fractions of malicious exit relays with exit policy that support this particular unpopular port could control large proportion of Tor traffics that need exit through the same particular unpopular port.

5.1. PROTOTYPE IMPLEMENTATION OF THE ATTACK

We developed the prototype implementation of the attack on a local machine using virtual box.

- First we developed a simple webpage using php-mysql technology hosted locally on apache web server. The webpage provide user with news or article and requires users to drop a feedback in a text area. This article webpage was poorly developed which makes it vulnerable to injection of a JavaScript.
- Apart from the webpage that provides article as described above, we also implemented a client and server sides, simple tcp application that is based on Websocket technology using socket.io. Socket.io is a websocket API. It is a node.js libarary which has ability to push traffic from client to a remote server through web browser proxy setting. So we installed both node.js and socket.io and later code a simple server side script that listen to port 6969. And client side script that recite on our apache web server directory. We tested the connection between the client side script that resides in our apache web server installation and the server side script that resides in Socket.IO installation. Client side script initiates connection to the remote server through port 6969.

- We compromised the article webpage by injecting a client side script which was then stored in the webpage. To observe the effect of the attack we require that each time a user visits the webpage a new tab containing the client side script is open and connected to the remote server.
- Finally, we visited the article webpage anonymously, using Tor browser bundle, and immediately a new tab opened by the side in the same Tor enabled web browser and established connection to the remote server which is listening to port 6969.
- This way we strongly believe that the Tor connection used to reach the article webpage (which passes through port 80) is likely to be different from the new open connection used to reach the remote server which listen to port 6969, since most exit relays exit policy do not support port 6969.

5.2. EVALUATING Tor PATH COMPROMISED DUE TO MALICIOUS ROUTERS EXITING UNPOPULAR PORTS

We obtained snapshot of the active onion routers from Tor's directory servers consisting of 2858 routers as of 27th March, 2012. We preprocessed the data to obtain information such as each router's name, status, version, self-advertised bandwidth and exit policy and

use these preprocessed data in our simulation, this is an indications of what client would experience while taking part in real Tor network.

To test the behavior of the above Tor path selection algorithm to unpopular ports, we firstly simulate the collected routers for some unpopular application without injecting any passive malicious routers. The applications are SMTP (25), NNTP (119), NNTP over SSL/TLS (563), Kazaa P2P (1214), Gnutella P2P (6346), Gnutella alternate (6347), eDonkey P2P (4661), BitTorrent (6881) and BitTorrent tracker (6969).

However, to evaluate how vulnerable is path compromise to unpopular ports malicious routers in the range of 8 to 112 are injected into the preprocessed dataset and used our developed path selection simulator to generates 1500 circuits for each of the above mention unpopular applications protocols (default ports numbers). Each malicious router consists of 10MB bandwidth, which is the maximum allowed bandwidth and advertises an exit policy that allows the client's application to exit only.

The same experiment was repeated with the different snapshot obtained from directory server on 14th April, 2012 consisting of 2998. However, in this case the numbers of circuits generated by path selection simulator are increased to 3000.

5.3. RESULTS

Since the focus of our work is the use of unpopular port to reveal the identity of client visiting a compromise server, the wide range of unpopular applications and their ports as listed above were simulated. By default these wide ranges of applications are rejected by Tor Network, partly because some of them leak information as they pass through the Tor network due to their unencrypting nature or when doing DNS lookup, and partly because some may carry viruses thereby exposing Tor relays to infection.

However, since Tor is an open source anonymity protocol in which individual users donates bandwidth to relay traffic, it is completely the responsibility of relay donors to decide how they intended to make contribution to Tor network. Some relay donors may decide to accept such unpopular ports in their exit policies. In this study we run Tor browser bundle and generate the statistic of Tor servers exiting unpopular ports as of snapshot of 1st March, 2012 in table 5.1. The table reveals how unpopular these ports are in Tor network, with NNTP over SSL (port 563) with highest number of servers (156 out of 2827 routers) ready to exit it, while the rest are insignificant; most of the amounts recorded are from small set of servers whose exit policy accept a range of port numbers that include most unpopular ports.

TABLE 5.1: Number of Tor Servers exit unpopular ports as of 1st March, 2012, at time interval between 9:00-12:00.

Port	**Number of Exit Nodes**
25	17
119	31
135-139	10
445	10
563	156
1214	10
4661-4666	13
6346-6420	10
6699	0
6881-6999	0

In another experiment we investigate the effects of injecting malicious exit routers to a normal Tor network by obtaining the counts for most frequently occurring router in total circuits generated for each application before adding malicious exit routers. And subsequently adding malicious exit routers in the ranges of 4 to 52 for the same application and recounts the number of times the same most frequently occurring router appears each time. Table 5.2 provides the relationship between a router that accept port 25 as its exit policy with the bandwidth of 559 (which has the highest occurrence in the circuits generated without injecting malicious routers) and the number of injected malicious exit routers with bandwidth of 10240. Simulation result shows that the router with the bandwidth of 559 appears 84 percent in 1500 circuits generated without injecting malicious exit routers. However, with only four malicious exit routers the percentage of exit router with the bandwidth 559 reduced by 55 in 1500 circuits generated. This trend is observed in all the remaining unpopular ports under investigation.

TABLE 5.2: percentage exit router with the highest bandwidth before and after injecting malicious exit routers compared to the % malicious exit router in the circuits

No of Malicious exit routers	**% malicious exit (bandwidth=10240)**	**% exit router without malicious(bandwidth=559)**
0	0	84.26666667
4	65.33333333	29.13333333
8	81.86666667	15.13333333
16	90.26666667	9.2
32	95.33333333	4.133333333
36	95.06666667	4.533333333
40	95.8	4.066666667
44	96.4	3.066666667
48	97.06666667	2.933333333
52	97.26666667	2.4
56	96.4	3.333333333

Finally, the results of path compromise rate obtained by simulating each of the nine unpopular ports mentioned above are shown in figure5.1 to figure5.9.

Path compromise rate indicates the percentage of the number of circuits in which malicious entry and malicious exit nodes appears, in other word the percentage of attack success in 1500 circuits generated for each port. It is observed from all figures below that there are fluctuations in the path compromise rate as the number of malicious router injected increases. This is due to random nature of router selection algorithm which sometimes may not favor router with higher bandwidth. However overall result shows that path compromise rate increases as the number of malicious routers injected increases in all unpopular ports.

Port 25 is officially used for email routing (SMTP) between mail servers. Figure 5.1 indicate that path compromise rate of 20 percent is the maximum obtained as the no of malicious routers increases to 112.

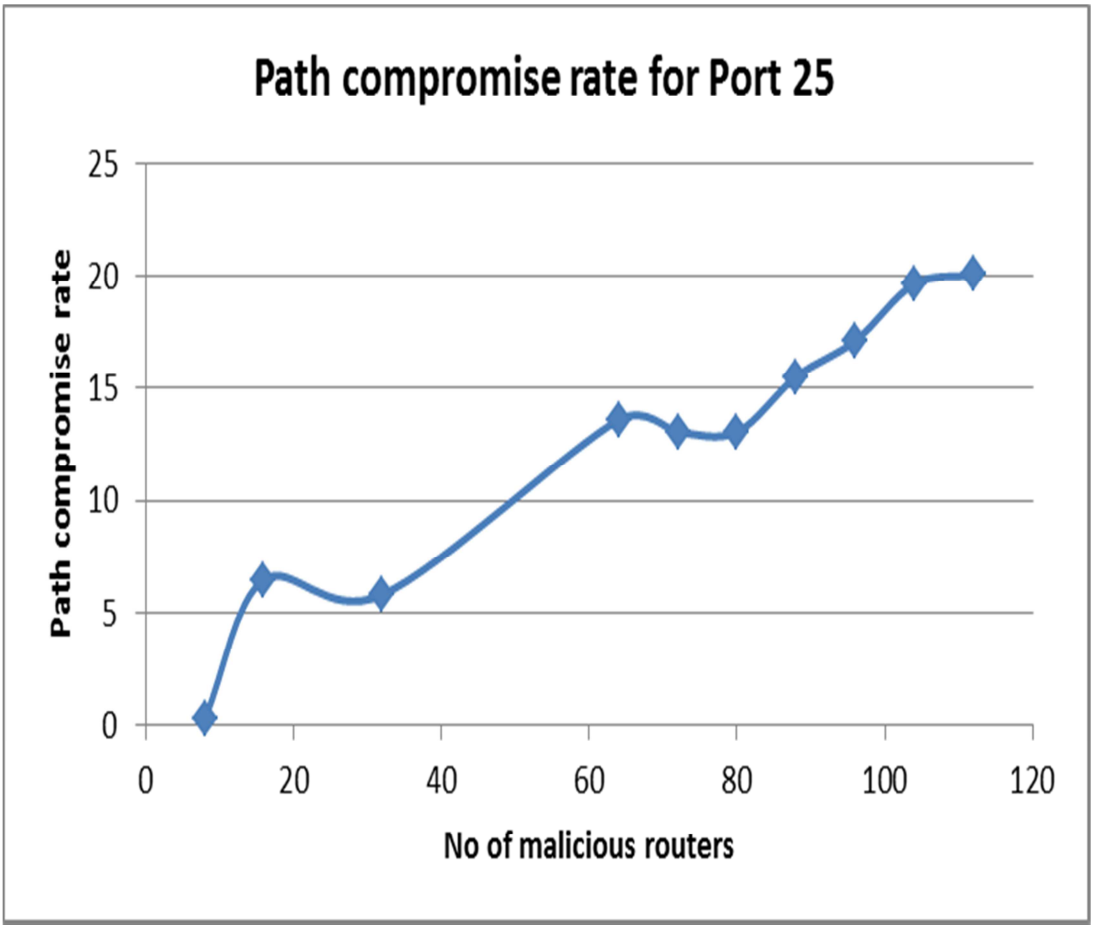


FIGURE 5.1: Path compromise rate for port 25 obtained from simulation of the default Tor path selection specification

Port 119 is officially used for retrieval of newsgroup messages (NNTP). The trend here indicates that path compromised rate increases as the number of malicious routers increases (see figure 5.2).

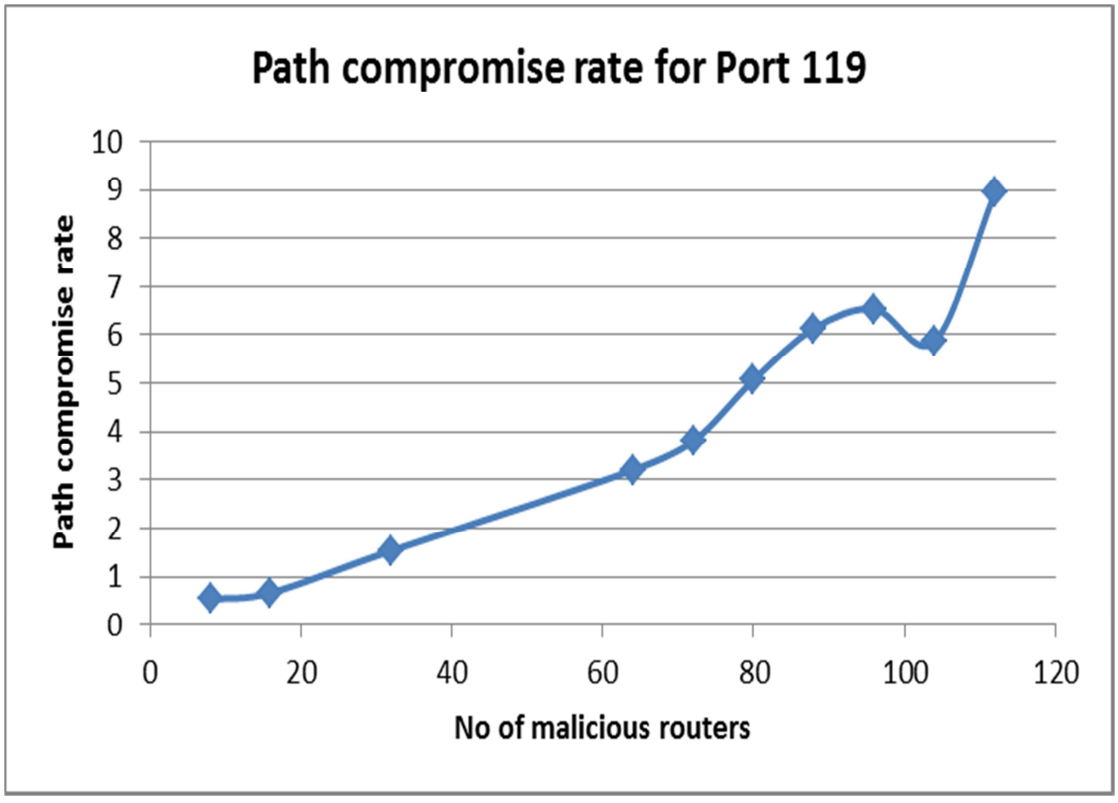


FIGURE 5.2: Path compromise rate for port 119 obtained from simulation of the default Tor path selection specification

Figure 5.3 shown the path compromise rate for port 563. This port support NNTP over SSL/TLS (NNTPS). Despite that port 563 indicates that as the number of malicious routers increases the path compromise rate also increases, it records the least compromise rate among the ports under consideration – approximately 8 percent as the number of malicious is 112.

Good to note that port 119 (see figure 5.2) which is the same protocol as port 563 (though unsecure) also records low compromise rate. The reason for these occurances could be explained by looking at the table 5.1 which shows that both ports have considerably large number of normal Tor routers that are willing to support such protocols in their exit policies. This implies that the chances of chosen malicious routers exiting such ports in Tor network will decrease significantly as indicated in both figures (5.2 and 5.3).

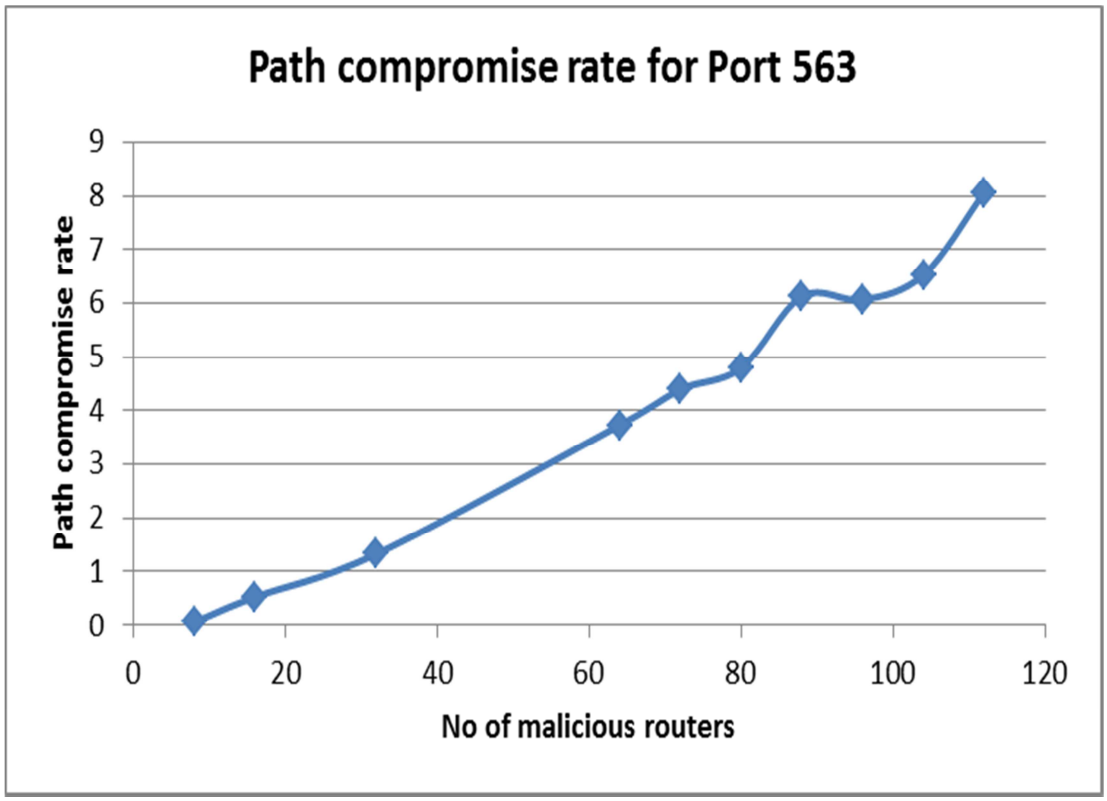


FIGURE 5.3: Path compromise rate for port 563 obtained from simulation of the default Tor path selection specification

Port 1214 is officially used by Kazaa (a peer-to-peer file sharing application). Figure 5.4 shown the path compromise rate increases steady as the number of malicious routers increases.

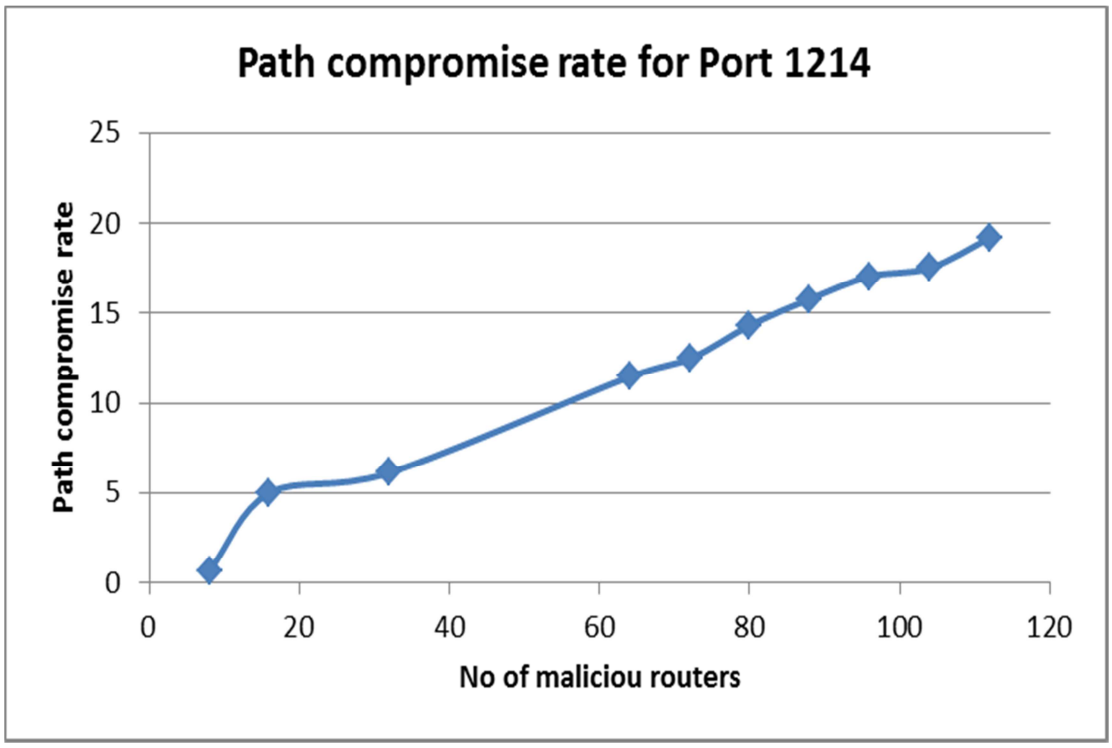


FIGURE 5.4: Path compromise rate for port 1214 obtained from simulation of the default Tor path selection specification

Port 4661 unofficially use by eDonky (a peer-to-peer application) has maximum path compromise rate of 20 percent as the number of malicious routers is 112 (see figure 5.5) and indicates increase in path compromise as the number of malicious routers increases.

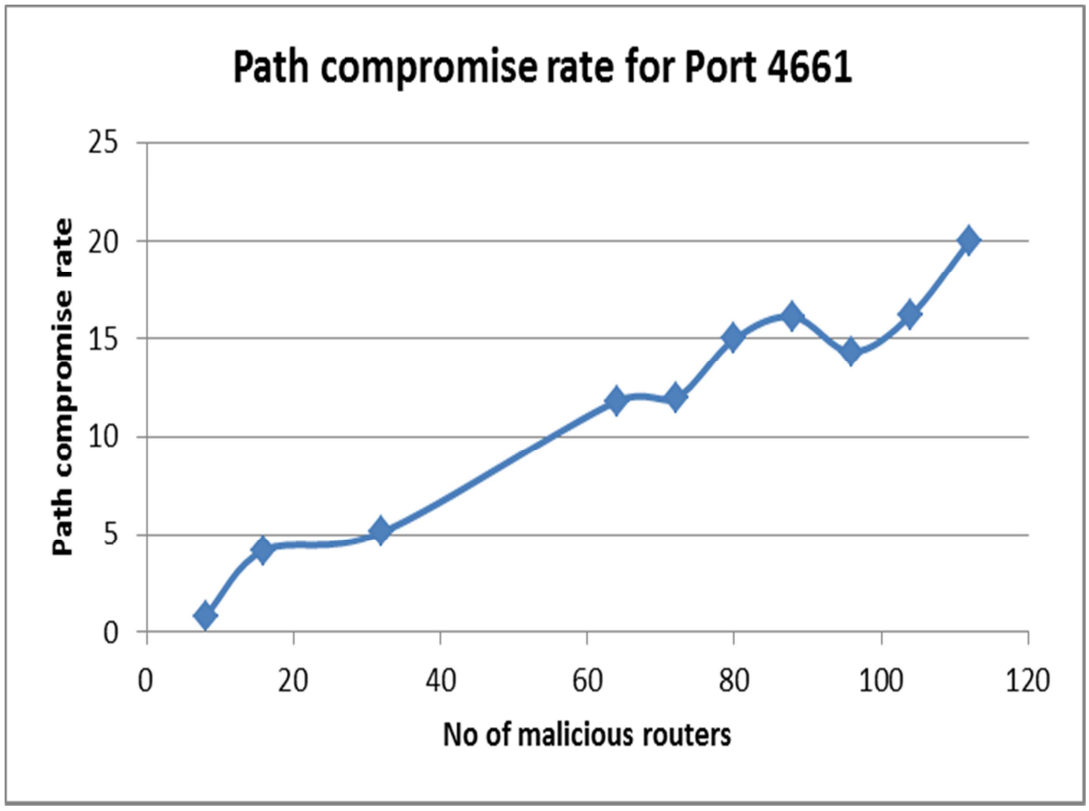


FIGURE 5.5: Path compromise rate for port 4661 obtained from simulation of the default Tor path selection specification

Port 6346 is officially used by gnutella (also a peer-to-peer applications). Figure 5.6 shown that the maximum path compromise is 18.6 percent at 104 malicious routers.

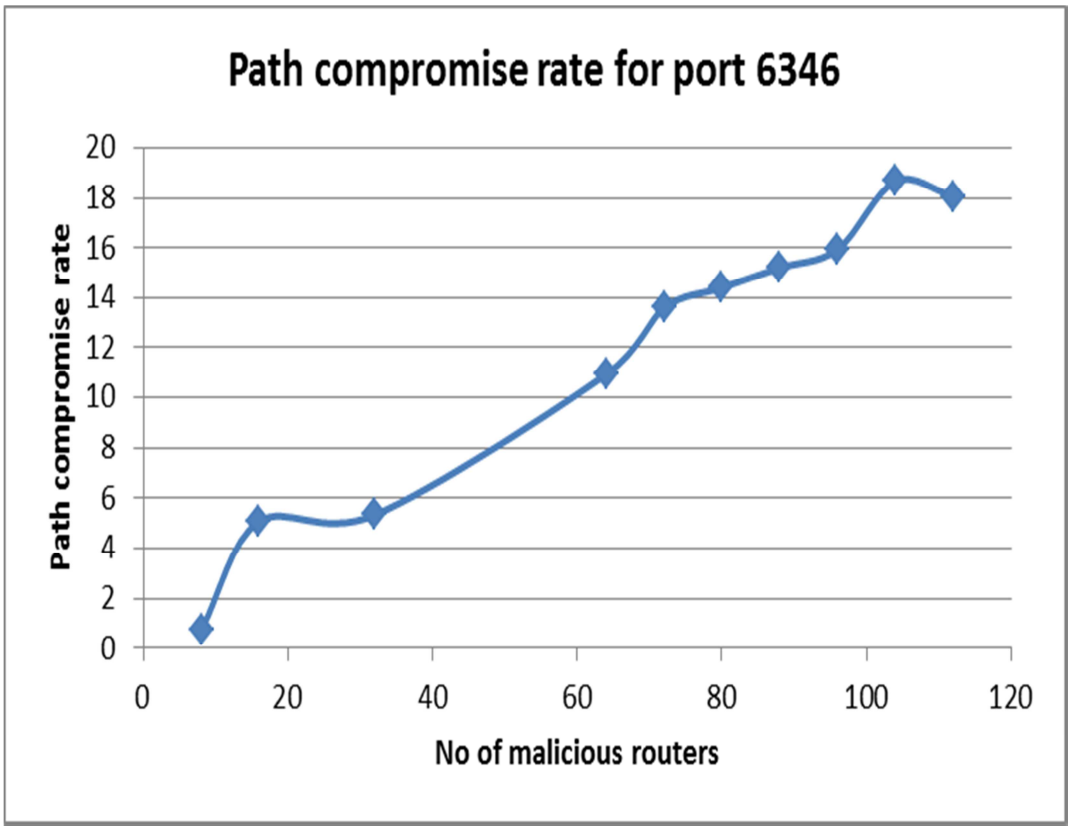


FIGURE 5.6: Path compromise rate for port 6346 obtained from simulation of the default Tor path selection specification

Port 6347 is officially use for gnutella alternate (a large peer-to-peer network) also for file sharing application. Maximum path compromise rate is 18 percent at 112 malicious routers (see figure 5.7)

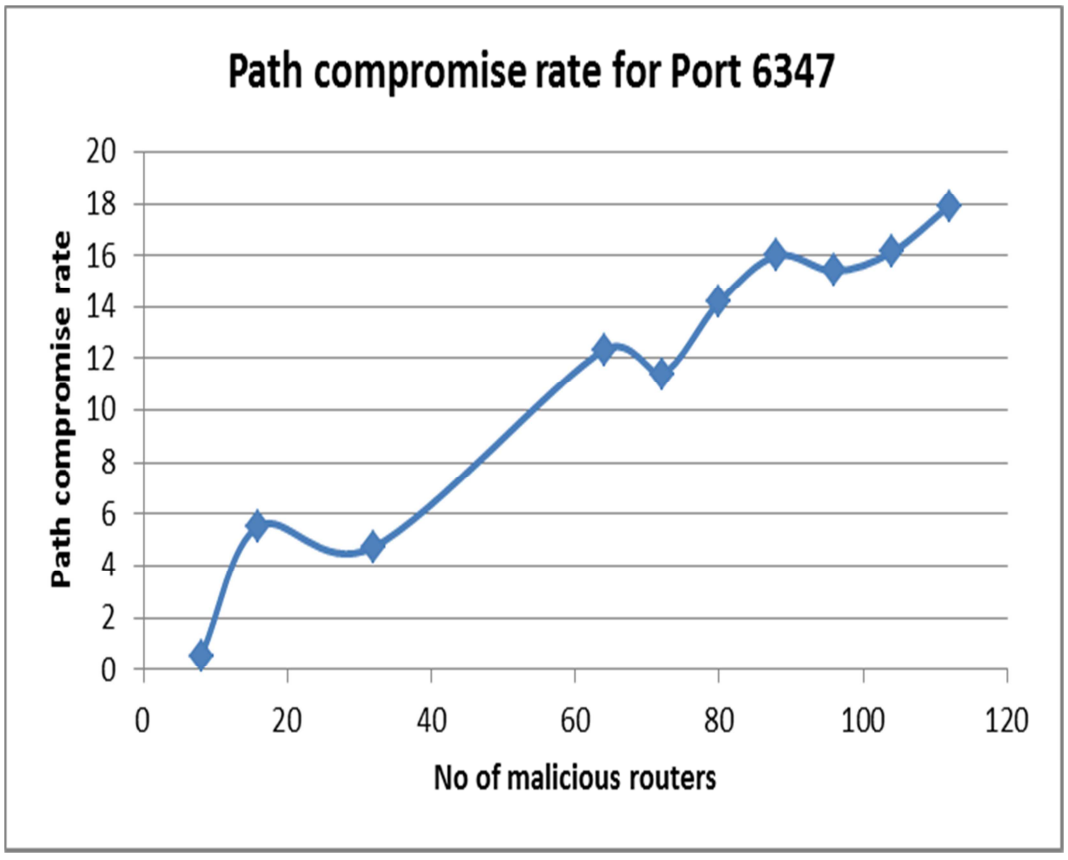


FIGURE 5.7: Path compromise rate for port 6347 obtained from simulation of the default Tor path selection specification

Figure 5.8 presents path compromise rate for port 6881 which is unofficially among the port used by BitTorrent. The maximum path compromise rate is roughly 18 percent obtained at 112 malicious routers.

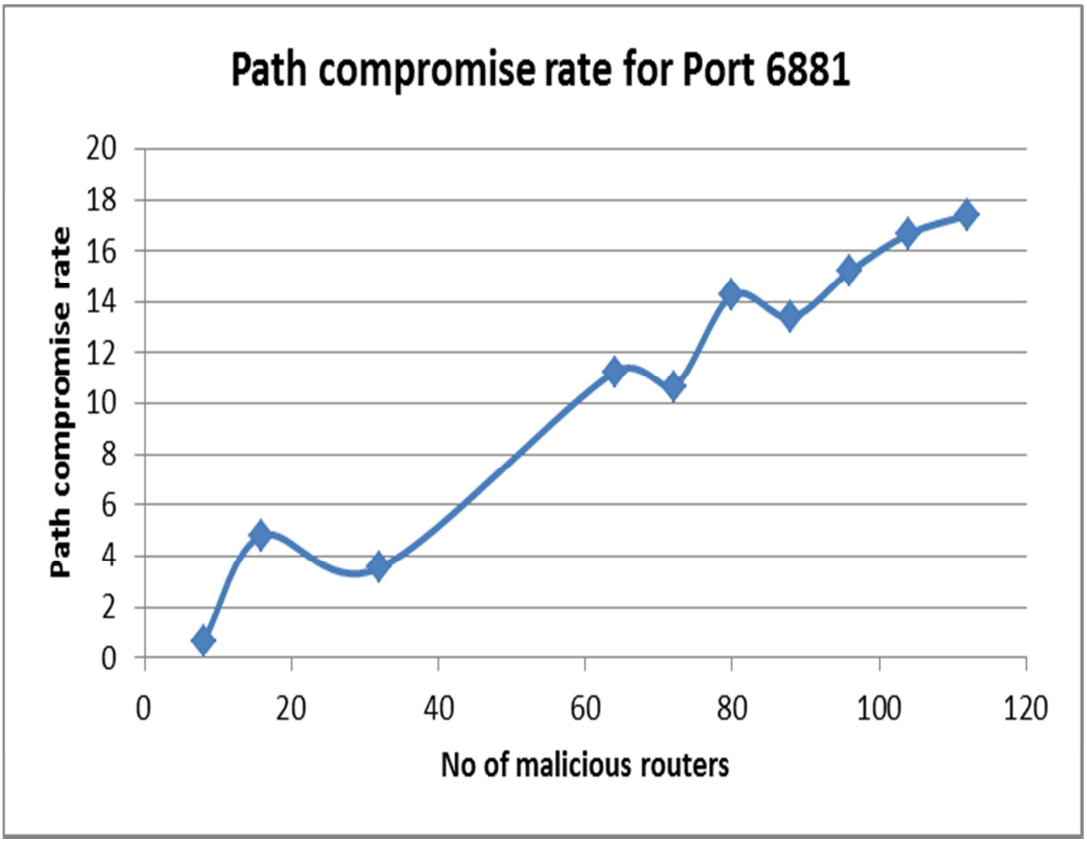


FIGURE 5.8: Path compromise rate for port 6881 obtained from simulation of the default Tor path selection specification

BitTorrent tracker unofficially uses port 6969 for end-to-end communication. Figure 5.9 presents path compromise rate against number of malicious routers, there is a steady increase in the path compromise rate as the number of malicious routers increases.

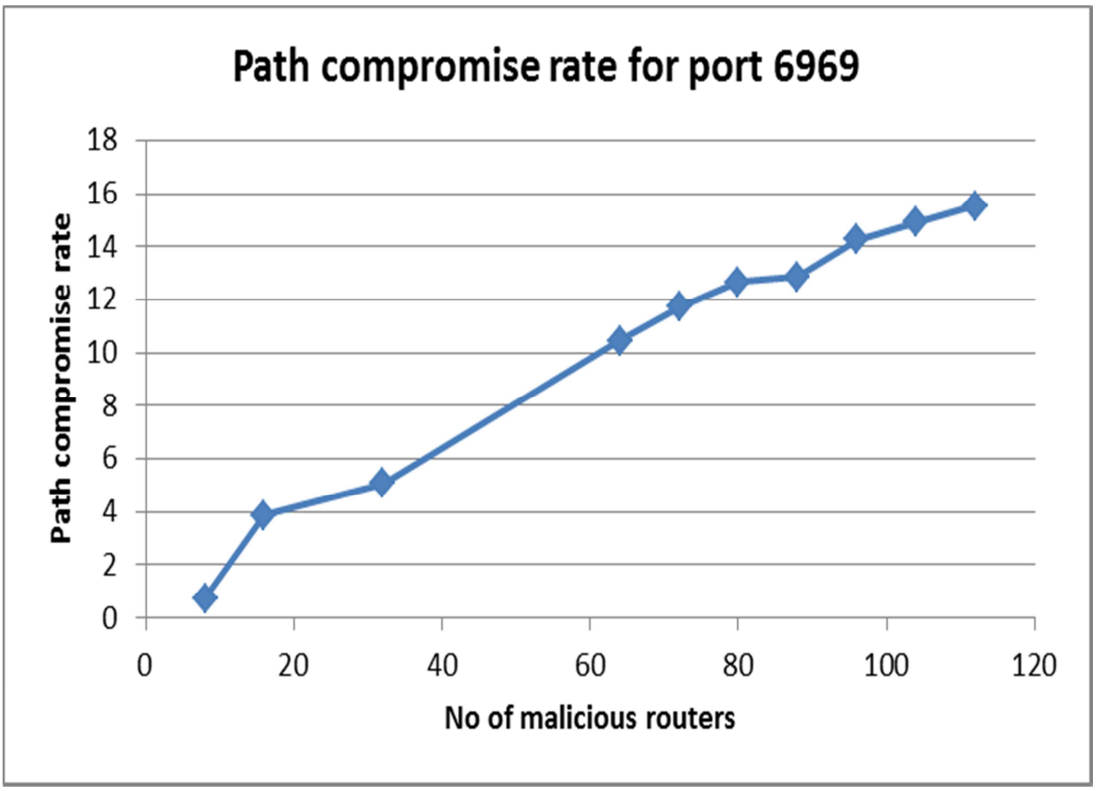


FIGURE 5.9: Path compromise rate for port 6969 obtained from simulation of the default Tor path selection specification

CHAPTER 6

ATTACK MITIGATION AND CONCLUSION

6.1. ATTACK MITIGATION

This type of attack is so strong that it is hard to be mitigated, since it involves injection of hidden script to a webpage unknown to both visitors and webserver (the host). However, certain number of precautions could be taken to mitigate the effect of the attack.

The first precaution is from webserver point of view. In this case ensuring that all inputs to webpages are validated by imposing validation rule during development of the site will prevent any external attacker from compromising the webpage, since compromising the webpage involve injecting a script on a webpage due to poor programming practice. So, this way the identity of anonymous visitor will remain unknown. However, in the case of internal attacker who is within the system and has appropriate privilege to edit a webpage

in a website. This type of attacker is extremely difficult to be detected since adding a line of hidden script on webpage will remain hidden to visitor and the host.

Regardless of the way a webserver is been compromised, anonymous client can mitigate against this attack by disabling all active objects (such as flash, JavaScript and active X) plugins on a web browser at least when using anonymity network like Tor. Most web browsers like Firefox provide users with option to disable any active contents from running when user visits a website that contains such embedded active objects. This way the chances of having hiding script to run on client machine is highly reduced. But this of course will come at a greater price, since most websites have useful active contents which when users disable all active plugins from running in their web browser this will affect user experience.

6.2. CONCLUSION

In this work we present a new attack idea which takes advantages of unpopular port to reveal anonymity of Tor clients visiting a compromised website. The attack involves forcing a client web browser to open a new connection that requires the use of an unpopular port in which the attacker's controlled set of Tor routers support.

We demonstrate the possibility of successfully implementing the attack by presenting different viable techniques, and we go further to present the prototype implementation of the attack.

Through simulation of Tor default path selection algorithm we are able to present the effect of injecting malicious router with considerably higher self-advertised bandwidth into Tor network, as well shown the probability of end-to-end attack on Tor network. Maximum of 20 percent compromise rate was recorded for some application ports as the number of malicious routers increases to 112. However, overall simulation results indicated that compromise rate increases as the number of injected malicious routers increases.

Finally, we provide different mitigation options against the attack.

APPENDIX A

A.1 Taxonomy of Free Anonymity Software

SN	Name	Platform	speed Rating	Anonymity Rating	Usage Allowance	Logging Level	Server location	OS	WOT Rating	RAM usage
1	JonDO	cross-platform	Slow	High	unlimited	Minimal	worldwide	Windows, Linux and Mac.	Excellent	76 MB
2	Vadalia	cross-platform	generally slow	medium - low	unlimited	varies	worldwide	Windows, Linux and Mac.	Excellent	44.5-52.5 MB
3	PocketiX.NET	cross-platform	medium	High	unlimited	high	Japan	Windows and Linux	Excellent	19MB
4	JanusVM	cross-platform	generally slow	High	unlimited	varies	worldwide	Windows and Linux	Excellent	53 MB
5	ProxPN	cross-platform	Fast	High	unlimited	Minimal	USA	Windows and Mac	Excellent	6.6 MB
6	USAIP	cross-platform	Fast	High	unlimited	Minimal	USA, UK, China, Netherland, and Germany	Windows, Linux and Mac.	unknown	

7	Your Freedom	cross-platform	Slow	High	Low	Minimal	worldwide	Windows, Linux and Mac.	Excellent	
8	xB Browser	Window-based	Slow	medium - low	unlimited	varies	worldwide	Windows	Excellent	
9	hotspot Shield	Window-based	Fast	High	Medium up to 5GB/Month	Minimal	USA and UK	Windows	Good	
10	AdvTor	Window-based	Slow	Medium-High	unlimited	varies	worldwide	Windows	Excellent	8 MB
11	SecurityKiss	Window-based	Fast	High	medium, 300 MB /day	Minimal	USA, UK, Canada, Switzerland, and Germany	Windows	unknown	5 MB
12	UltraSurf	Window-based	Fast	High	unlimited	high	Usa	Windows	Good	5 MB
13	CyberGhost VPN	Window-based	Fast	High	low 1 Gb/month	unknown	Germany	Windows	Excellent	27 MB
14	FreeGate	Window-based	medium	Medium	unlimited	unknown	worldwide	windows	Excellent	15 MB
15	Incognito Live System	Linux-based	medium	High	unlimited	varies	worldwide	Linux	Good	
16	Estrela Roja	Linux-based	Slow	Medium-High	unlimited	varies	worldwide	Linux	unknown	
17	Privatix	Linux-based	Slow	Medium-High	unlimited	varies	worldwide	Linux	Excellent	

APPENDIX B

B.1 Experimental Results for 1500 circuits generated.

Table 4: simulation Result for Port 25 with malicious

No of Malicious routers	**total bandwidth in MB**	**no of malicious exit**	**no of malicious entry**	**no of match**	**total malicious**	**% match**	**% of malicious**	**% malicious exit**
8	80	980	8	4	984	0.27	65.60	65.33
16	160	1228	116	97	1247	6.47	83.13	81.87
32	320	1354	98	87	1365	5.80	91.00	90.27
64	640	1430	214	204	1440	13.60	96.00	95.33
72	720	1426	203	196	1433	13.07	95.53	95.07
80	800	1437	205	196	1446	13.07	96.40	95.80
88	880	1446	247	233	1460	15.53	97.33	96.40
96	960	1456	261	257	1460	17.13	97.33	97.07
104	1040	1459	299	295	1463	19.67	97.53	97.27
112	1120	1446	310	301	1455	20.07	97.00	96.40

Table 5: simulation Result for Port 119 with malicious

No of Malicious routers	**total bandwidth in MB**	**no of malicious exit**	**no of malicious entry**	**no of match**	**total malicious**	**% match**	**% of malicious**	**% malicious exit**
8	80	65	8	8	65	0.53	4.33	4.33
16	160	144	120	10	254	0.67	16.93	9.60
32	320	265	114	23	356	1.53	23.73	17.67
64	640	421	181	48	554	3.20	36.93	28.07
72	720	489	209	57	641	3.80	42.73	32.60
80	800	509	258	76	691	5.07	46.07	33.93
88	880	573	234	92	715	6.13	47.67	38.20
96	960	571	265	98	738	6.53	49.20	38.07
104	1040	560	254	88	726	5.87	48.40	37.33
112	1120	634	296	134	796	8.93	53.07	42.27

Table 6: simulation Result for Port 563 with malicious

No of Malicious routers	**total bandwidth in MB**	**no of malicious exit**	**no of malicious entry**	**no of match**	**total malicious**	**% match**	**% of malicious**	**% malicious exit**
8	80	72	38	1	109	0.07	7.27	4.80
16	160	158	133	8	283	0.53	18.87	10.53
32	320	251	106	20	337	1.33	22.47	16.73
64	640	430	210	56	584	3.73	38.93	28.67
72	720	453	232	66	619	4.40	41.27	30.20
80	800	497	219	72	644	4.80	42.93	33.13
88	880	535	238	92	681	6.13	45.40	35.67
96	960	569	260	91	738	6.07	49.20	37.93
104	1040	564	274	98	740	6.53	49.33	37.60
112	1120	645	301	121	825	8.07	55.00	43.00

Table 7: simulation Result for Port 1214 with malicious

No of Malicious routers	**total bandwidth in MB**	**no of malicious exit**	**no of malicious entry**	**no of match**	**total malicious**	**% match**	**% of malicious**	**% malicious exit**
8	80	641	37	10	668	0.67	44.53	42.73
16	160	939	126	75	990	5.00	66.00	62.60
32	320	1171	114	92	1193	6.13	79.53	78.07
64	640	1314	198	172	1340	11.47	89.33	87.60
72	720	1328	214	187	1355	12.47	90.33	88.53
80	800	1353	247	215	1385	14.33	92.33	90.20
88	880	1341	267	237	1371	15.80	91.40	89.40
96	960	1347	280	256	1371	17.07	91.40	89.80
104	1040	1395	285	263	1417	17.53	94.47	93.00
112	1120	1386	302	288	1400	19.20	93.33	92.40

Table 8: simulation Result for Port 4661 with malicious

No of Malicious routers	**total bandwidth in MB**	**no of malicious exit**	**no of malicious entry**	**no of match**	**total malicious**	**% match**	**% of malicious**	**% malicious exit**
8	80	639	34	13	660	0.87	44.00	42.60
16	160	940	111	63	988	4.20	65.87	62.67
32	320	1160	108	77	1191	5.13	79.40	77.33
64	640	1289	202	177	1314	11.80	87.60	85.93
72	720	1331	211	179	1363	11.93	90.87	88.73
80	800	1362	244	225	1381	15.00	92.07	90.80
88	880	1353	264	242	1375	16.13	91.67	90.20
96	960	1357	242	215	1384	14.33	92.27	90.47
104	1040	1385	262	243	1404	16.20	93.60	92.33
112	1120	1378	326	300	1404	20.00	93.60	91.87

Table 9: simulation Result for Port 6346 with malicious

No of Malicious routers	**total bandwidth in MB**	**no of malicious exit**	**no of malicious entry**	**no of match**	**total malicious**	**% match**	**% of malicious**	**% malicious exit**
8	80	595	42	11	626	0.73	41.73	39.67
16	160	944	115	76	983	5.07	65.53	62.93
32	320	1136	109	80	1165	5.33	77.67	75.73
64	640	1293	191	164	1320	10.93	88.00	86.20
72	720	1320	230	204	1346	13.60	89.73	88.00
80	800	1348	237	216	1369	14.40	91.27	89.87
88	880	1344	260	228	1376	15.20	91.73	89.60
96	960	1372	254	239	1387	15.93	92.47	91.47
104	1040	1352	315	280	1387	18.67	92.47	90.13
112	1120	1384	296	271	1409	18.07	93.93	92.27

Table 10: simulation Result for Port 6347 with malicious

No of Malicious routers	**total bandwidth in MB**	**no of malicious exit**	**no of malicious entry**	**no of match**	**total malicious**	**% match**	**% of malicious**	**% malicious exit**
8	80	631	30	8	653	0.53	43.53	42.07
16	160	952	135	83	1004	5.53	66.93	63.47
32	320	1135	95	71	1159	4.73	77.27	75.67
64	640	1311	214	185	1340	12.33	89.33	87.40
72	720	1327	196	171	1352	11.40	90.13	88.47
80	800	1343	241	213	1371	14.20	91.40	89.53
88	880	1347	266	240	1373	16.00	91.53	89.80
96	960	1352	259	231	1380	15.40	92.00	90.13
104	1040	1374	269	242	1401	16.13	93.40	91.60
112	1120	1378	296	269	1405	17.93	93.67	91.87

Table 11: simulation Result for Port 6881 with malicious

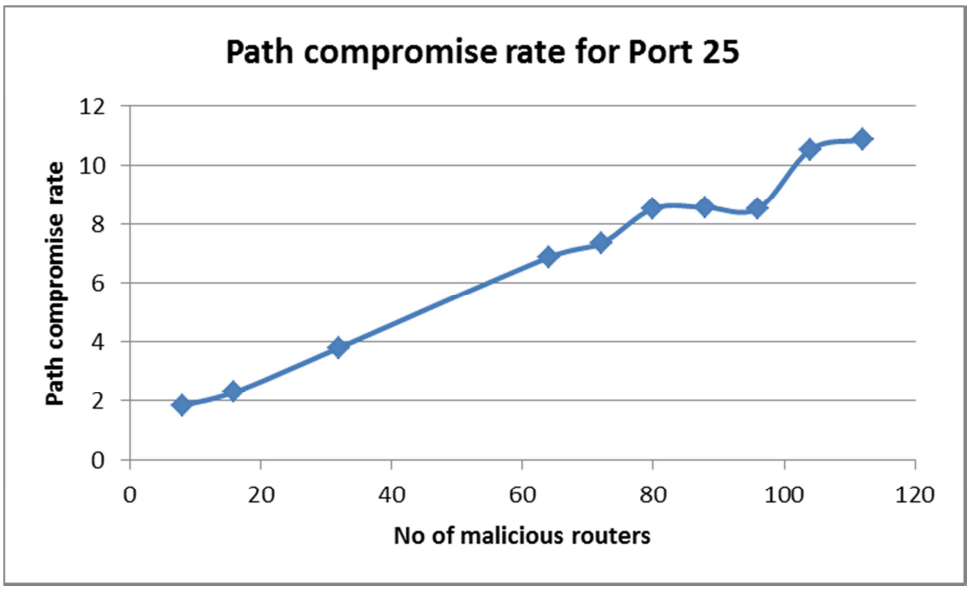
No of Malicious routers	**total bandwidth in MB**	**no of malicious exit**	**no of malicious entry**	**no of match**	**total malicious**	**% match**	**% of malicious**	**% malicious exit**
8	80	431	39	10	460	0.67	30.67	28.73
16	160	757	128	72	813	4.80	54.20	50.47
32	320	975	79	53	1001	3.53	66.73	65.00
64	640	1208	217	169	1256	11.27	83.73	80.53
72	720	1217	198	160	1255	10.67	83.67	81.13
80	800	1229	265	214	1280	14.27	85.33	81.93
88	880	1277	238	201	1314	13.40	87.60	85.13
96	960	1281	266	228	1319	15.20	87.93	85.40
104	1040	1294	290	250	1334	16.67	88.93	86.27
112	1120	1333	288	261	1360	17.40	90.67	88.87

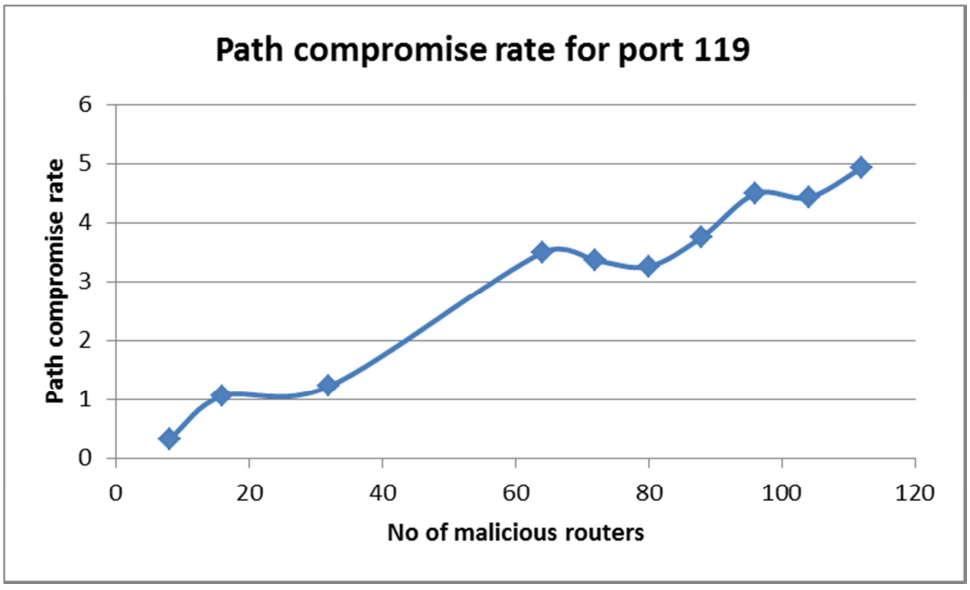
Table 12: simulation Result for Port 6969 with malicious

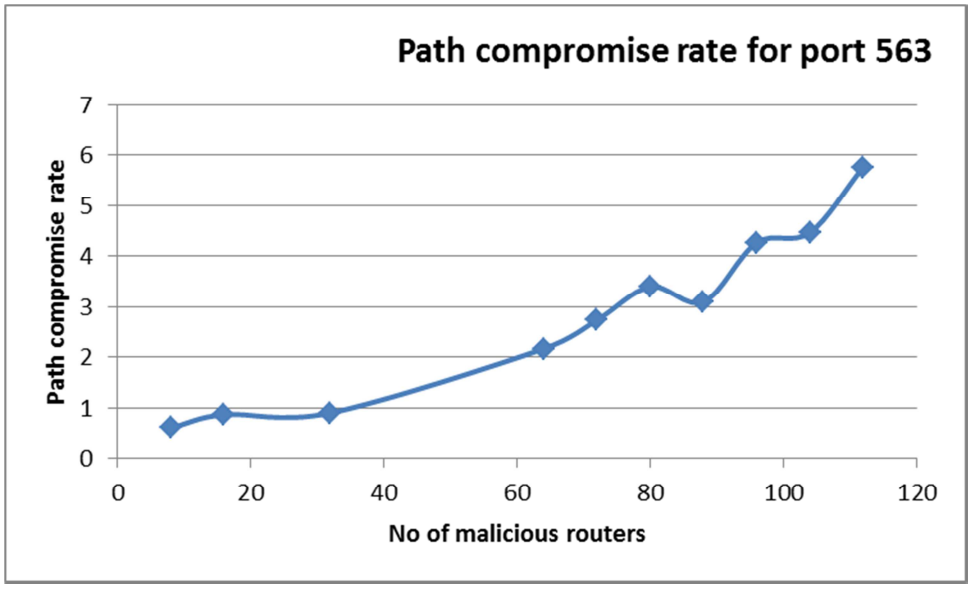
No of Malicious routers	**total bandwidth in MB**	**no of malicious exit**	**no of malicious entry**	**no of match**	**total malicious**	**% match**	**% of malicious**	**% malicious exit**
8	80	414	45	11	448	0.73	29.87	27.60
16	160	682	118	58	742	3.87	49.47	45.47
32	320	892	108	76	924	5.07	61.60	59.47
64	640	1127	206	157	1176	10.47	78.40	75.13
72	720	1149	216	176	1189	11.73	79.27	76.60
80	800	1171	243	190	1224	12.67	81.60	78.07
88	880	1198	265	193	1270	12.87	84.67	79.87
96	960	1219	263	214	1268	14.27	84.53	81.27
104	1040	1231	290	224	1297	14.93	86.47	82.07
112	1120	1237	289	233	1293	15.53	86.20	82.47

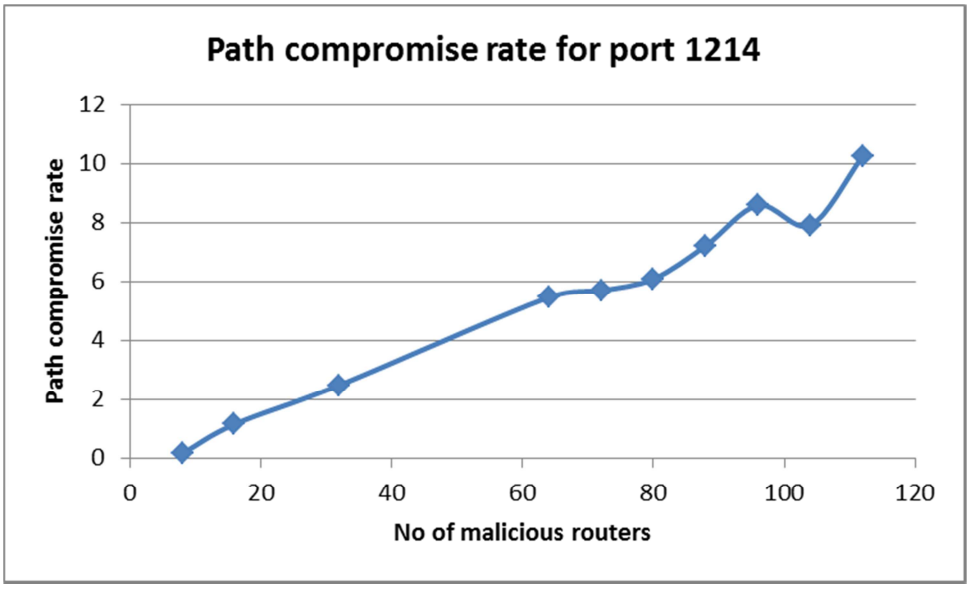
APPENDIX C

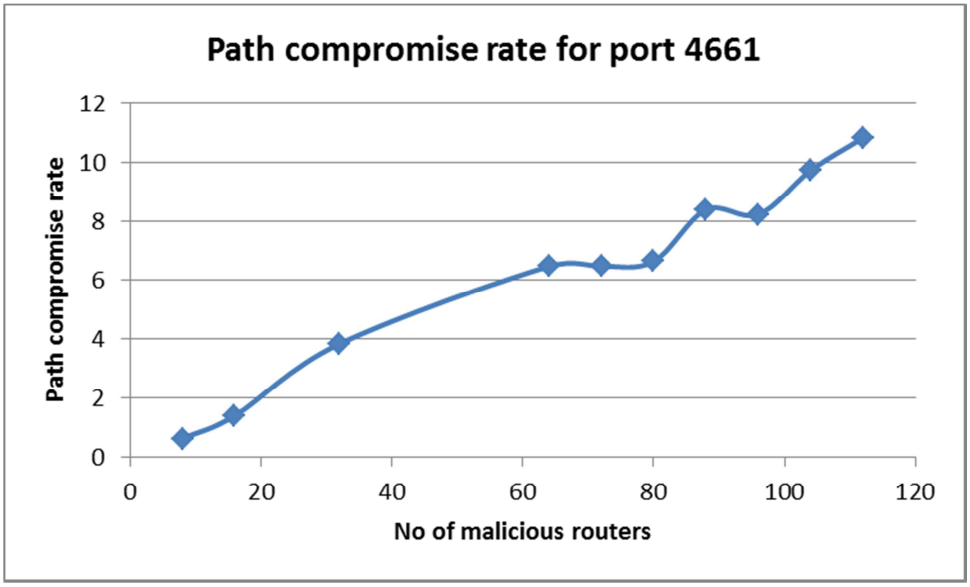
C.1 Figures generated in simulating path selection for 3000 circuits

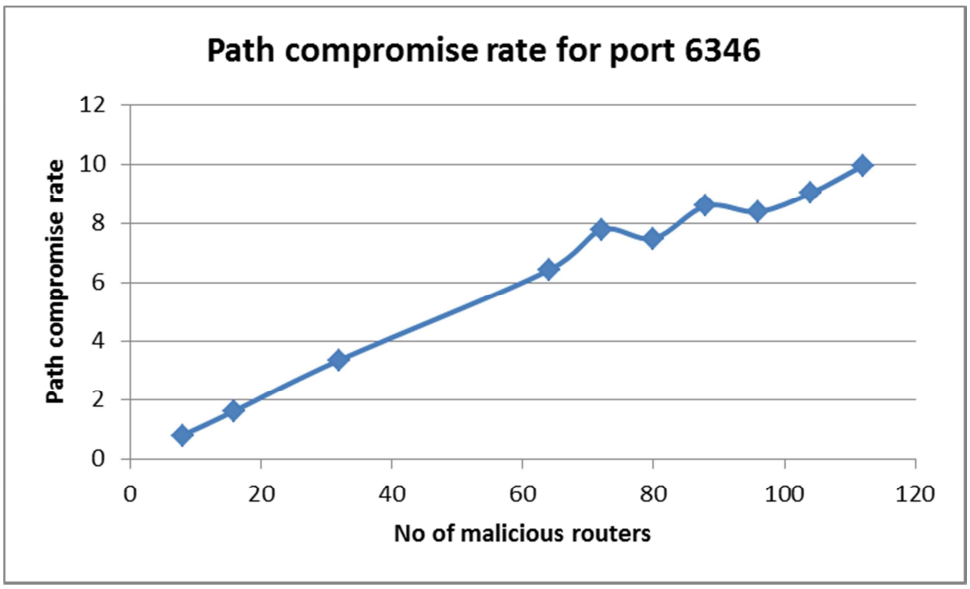


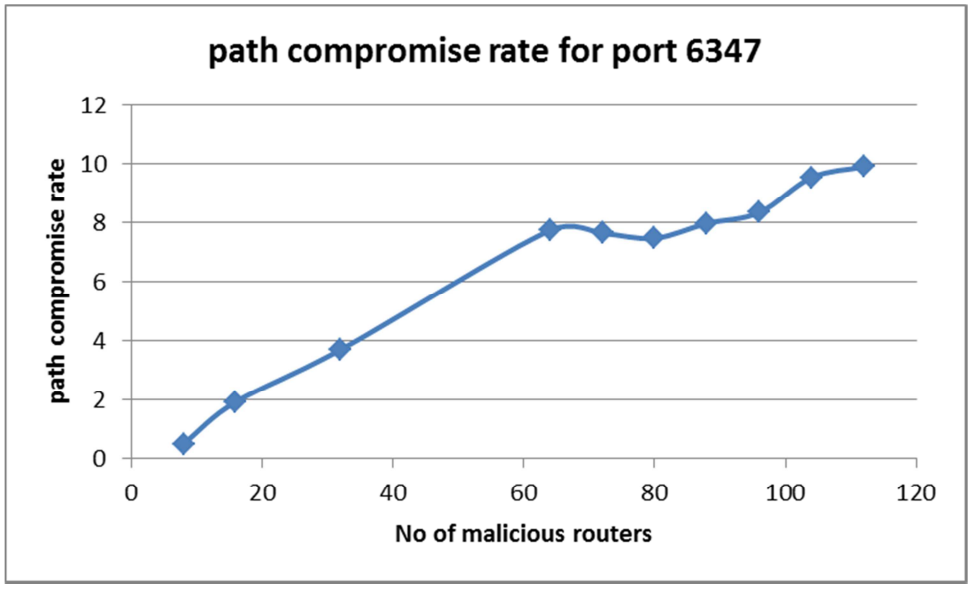


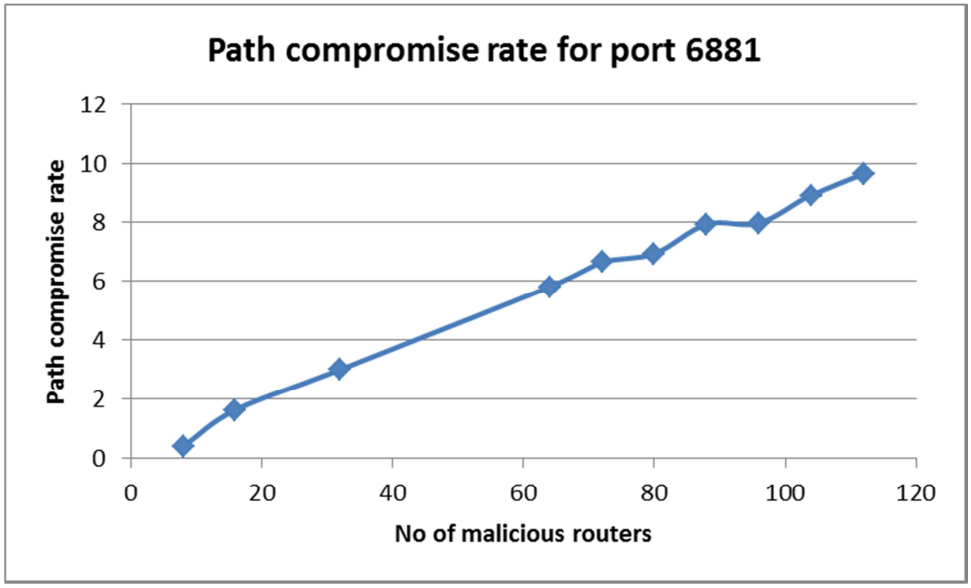

Path compromise rate for port 563
Path compromise rate
0
1
2
3
4
5
6
7
0
20
40
60
80
100
120
No of malicious routers

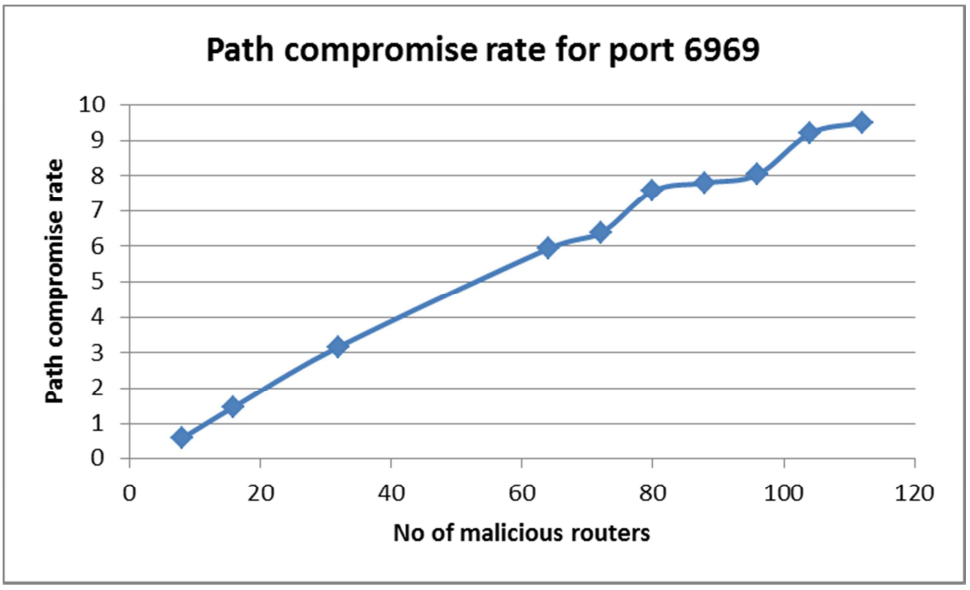

Path compromise rate for port 1214
Path compromise rate
0
2
4
6
8
10
12
0
20
40
60
80
100
120
No of malicious routers


Path compromise rate for port 4661
Path compromise rate
0
2
4
6
8
10
12
0
20
40
60
80
100
120
No of malicious routers


Path compromise rate for port 6346
Path compromise rate
0
2
4
6
8
10
12
0
20
40
60
80
100
120
No of malicious routers


path compromise rate for port 6347
path compromise rate
12
10
8
6
4
2
0
0
20
40
60
80
100
120
No of malicious routers


Path compromise rate for port 6881
Path compromise rate
12
10
8
6
4
2
0
0
20
40
60
80
100
120
No of malicious routers


Path compromise rate for port 6969
Path compromise rate
10
9
8
7
6
5
4
3
2
1
0
0
20
40
60
80
100
120
No of malicious routers

REFERENCES

[1] George Danezis, Claudia Diazy and Paul Syversonz, Systems for Anonymous Communication August 31, 2009.

[2] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, Low-Resource Routing Attacks against Tor, In the Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007), Washington, DC, USA, October 2007.

[3] Tor: Overview, TorProject.org; https://www.torproject.org/about/overview.htm

[4] Roger Dingledine, and Nick Mathewson, Tor Protocol Specification; https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=tor-spec.txt

[5] Kevin Bauer, Dirk Grunwald, and Douglas Sicker, Predicting Tor Path Compromise by Exit Port, 2nd IEEE International Workshop on Information and Data Assurance (WIDA) Phoenix, USA, December 2009.

[6] Steven J. Murdoch and Robert N. M. Watson, Metrics for Security and Performance in Low-Latency Anonymity Networks. In the Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008), Leuven, Belgium, July 2008, pages 115-132.

[7] Chaum, D., Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of ACM 24(2) (1981) 84–90.

[8] Chaum, D., The dining Cryptographers' problem: unconditional sender and recipient untraceability. J. Cryptol. 1(1) (1988) 65–75.

[9] Reiter, M., Rubin, A., Crowds: Anonymity for web transactions, ACM Transactions on Information System Security, Vol. 1, and No. 1. (November 1998), pp. 66-92.

[10] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong, Freenet: A Distributed Anonymous Information Storage and Retrieval System, Proceedings of Designing Privacy Enhancing Technologies, 2001.

[11] Shields, C., Levine, B.: A protocol for anonymous communication over the internet. In: Proceedings of the 7th ACM Conference on Computer and Communications Security, New York, NY, USA, ACM (2000) 33–42.

[12] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router.In 13th USENIX Security Symposium (2004).

[13] M. J. Freedman and R. Morris, Tarzan: A peer-to-peer anonymizing network layer, In Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, DC, November 2002.

[14] Marta Rybczy´nska, Network-level properties of modern anonymity systems, Proceedings of the International Multiconference on Computer Science and Information Technology pp. 837–843.

[15] John Sullivan, Executive Director, Free Software Foundation, 2010 Free Software Awards Announced; Published on March 22, 2011.

http://www.fsf.org/news/2010-free-software-awards-announced

[16] M. Wright, M. Adler, B. Levine, and C. Shields; an analysis of Degradation of Anonymous Protocols.

[17] Pfitzmann and Pfitzmann, How to Break the Direct RSA-Implementation of Mixes, in the Proceedings of EUROCRYPT '89, LNCS, Springer-Verlag, Berlin 1990.

[18] RSA - Wikipedia, the free encyclopaedia: http://en.wikipedia.org/wiki/RSA

[19] Murdoch and Danezis, Low-Cost Traffic Analysis of Tor, 2005 IEEE Symposium on Security and Privacy, May 8–11 2005, Oakland, California, USA.

[20] Steven J. Murdoch and Piotr Zieli´nski, Sampled Traffic Analysis by Internet-Exchange-Level Adversaries, University of Cambridge, Computer Laboratory

http://www.cl.cam.ac.uk/users/ {sjm217, pz215}

[21] Overlier, L. and Syverson P., Locating Hidden Servers; In Proceedings of the IEEE Symposium on Security and Privacy. 2006.

[22] Murdoch J. Steven, Hot or Not: Revealing Hidden Services by their Clock Skew, 13th ACM Conference on Computer and Communications Security (CCS), Alexandria, Virginia.

[23] Hotspot Shield VPN: http://www.hotspotshield.com

[24]. Timothy G, Abbott, Katherine J. Lai, Michael R. Lieberman, and Eric C. Price, Browser-Based Attacks on Tor, In the Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007), Ottawa, Canada, June 2007.

[25] Elices, Juan A Perez-Gonzalez, Fernando Troncoso, Carmela, Fingerprinting Tor's Hidden Service Log Files Using a Timing Channel, IEEE International Workshop on Information Forensics and Security, 2011.

[26] Ling, Zhen Luo, Junzhou Yu, Wei Fu, Xinwen Xuan, Dong Jia, Weijia, A Cell-Counting-Based Attack against Tor, Proceedings of the 16th ACM conference on Computer and communications security CCS 2009.

[27] Pries, R Yu, W Fu, X Zhao, W, A New Replay Attack against Anonymous Communication Networks, IEEE International Conference on Communications, 2008.

[28] Panchenko, Andriy Renner, Johannes, Path selection metrics for Performance-Improved Onion Routing, Ninth Annual International Symposium on Applications and the Internet, 2009.

[29] Loesing, Karsten; Murdoch, Steven J; Dingledine, Roger, a case Study on Measuring Statistical Data in the Tor Anonymity Network, Workshop on Ethics in Computer Security Research (WECSR 2010), January 2010 .

[30] Danner, Norman; Krizanc, Danny; Liberatore, Marc; Detecting DOS Attacks in Tor, Financial Cryptography and Data Security, Lecture Notes in Computer Science, Volume 5628. ISBN 978-3-642-03548-7, Springer Berlin Heidelberg, 2009, p. 273

[31] Liu, Xin Wang, Neng, Anti-misbehavior System for Tor Network, INC IMS and IDC 2009 NCM 09 Fifth International Joint Conference.

[32] Evans, Nathan S; Dingledine, Roger; Grothoff, Christian, A practical Congestion Attack on Tor Using Long Path, 18th USENIX Security Symposium, 2009.

[33] Wang, Xiaogang Luo, Junzhou Yang, Ming Ling, Zhen; A Potential HTTP-based application-level attack against Tor, Future Generation Computer Systems Volume 27, Issue 1, January 2011, Pages 67–77.

[34] Bauer, Kevin Juen, Joshua Borisov, Nikita Grunwald, Dirk Sicker, Douglas McCoy, Damon; On the Optimal Path Length for Tor, 10th Privacy Enhancing Technologies Symposium in Berlin, Germany, July 2010.

[35] Manils, Pere Abdelberri, Chaabane Blond, Stevens Le Kaafar, Mohamed Ali Castelluccia, Claude Legout, Arnaud Dabbous, Walid, Compromising Tor Anonymity Exploiting P2P Information Leakage, Networking and Internet Architecture (cs.NI); Cryptography and Security (cs.CR), arXiv:1004.1461v1 [cs.NI], 2010.

[36] Jeremiah Grossman, WhiteHat Website Security Statistics Report, 2007

[37] WebSocket, Wikipedia; http://en.wikipedia.org/wiki/WebSocket

[38] David Walsh Blog; WebSocket and Socket.IO, http://davidwalsh.name/websocket

[39] David Fifield; WebSocket Pluggable Transport for Tor;

http://archives.seul.org/or/dev/Apr-2012/msg00019.html

VITA

Name: Muhammad Aliyu Sulaiman

Date of Birth: 19th January, 1980

Nationality: Nigeria.

- Recieved first degree in Computer Scinece (B.Sc. Hons) from Kano University of Science and Technology, Wudil, 2006.
- Completed Master of Scinece degree in Computer Science from King Fahd University of Petroleum and Minerals, Dhahran, Kingdom of Saudi Arabia, 2012.
- My email contact is muhalisu@gmail.com