University of Nevada, Reno

An Assessment of Current Attacks on Anonymous Networks

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

by

Christopher R. Zachor

Dr. Mehmet Hadi Gunes / Thesis Advisor

Dr. George Bebis/ Thesis Advisor

May, 2012

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

CHRISTOPHER R. ZACHOR

entitled

An Assessment Of Current Attacks On Anonymous Networks

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Dr. Mehmet Gunes, Advisor

Dr. George Bebis, Committee Member

Dr. Emmanuel Barthe, Graduate School Representative

Marsha H. Read, Ph. D., Dean, Graduate School

May, 2012

Abstract

Anonymous networks are important for maintaining ones privacy on the internet. However, these networks can be used for both good and bad. While a humanitarian worker may use one to spread the word about atrocities committed by an oppressive regime, a person with malicious intent might use it to compromise a system anonymously. To this extent, it is important for law enforcement to understand anonymous networks and how they can be countered in a criminal investigation. Many attacks against networks such as Tor already exist. Some of these attacks require resources inaccessible to the average investigator. This thesis attempts to identify mechanisms that law enforcement can utilize to track cyber criminals that use anonymous networks.

Acknowledgements

This research project was funded by the United States Department of Justice through a grant from the National Institute of Justice #2010-DN-BX-K248.

I would like to start by thanking my advisors. Dr. Mehmet Gunes has helped me better understand the world of research. He helped me get my first paper published and has given me advice that will help with any future problem I might try to solve. Dr. George Bebis who guided me through the process of earning my degree. I would also like to thank my final committee member, Dr. Emmanuel Barthe from the Criminal Justice Department, for taking time from his schedule to make this possible.

I would also like to thank Todd and Linda Shipley of Great Basin Data Recovery for allowing me to hang out at their shop and learn all I could about digital forensics. And last but not least, I would like to thank my family. My Mom and Dad have always encouraged and supported me. I always knew what I wanted to do and they always allowed me the freedom to figure out how I wanted to get there. And thank you to my brother and my sister for always putting up with me. To all of you, I offer my sincerest thank you.

Christohpher R. Zachor

Table of Contents

Table of Figures

Chapter 1

Introduction

Computer security has always been a game of cat and mouse. The "bad" guys will develop exploits to compromise computer systems while the "good" guys will try to stop them (the good guys also develop exploits, but for the purpose of securing systems). Often times, the good guys will develop tools to test the security of systems [1]. However, as with many things in our daily lives, these tools can be used for both good and evil. This scenario is also true for anonymizing networks. Privacy enhancing networks can be used by anyone no matter what color hat they wear.

Law enforcement and security researchers use these networks to remain anonymous while investigating websites [2]. Leaving a government IP address or your business' IP address in the logs of a system under investigation can tip the bad guy off or leave the investigator vulnerable for retaliation. Human rights groups have also found anonymizing networks useful for communicating with the outside world under the watchful eye of an oppressive regime. Of course with the good uses, there are always those who will use these technologies to hide their criminal activity.

One such case occurred in Washington State [3]. A student (identity unknown at the time of his threats) used an anonymously created myspace.com account to repeatedly send

bomb threats to his school, thus, disrupting classes. The local police force called the FBI for assistance in tracking the individual down. It was later revealed through the affidavit for a search warrant that the FBI has a tool for this type of job. The Computer and Internet Protocol Address Verifier, simply known as CIPAV, acts much like malware seen in the wild today. It can collect information necessary to fingerprint targeted systems despite the operator's use of anonymizing technologies. The full list of its capabilities is still unknown as is the method or methods of delivery. Security researchers have speculated on the avenues of attack, but it is likely that only a handful of people and the FBI know for sure. One such method proposed was the Microsoft Animated Cursor vulnerability [4]. While another avenue proposed was vulnerability in Microsoft's Graphics Rendering Engine [5] which was used to target myspace.com users previously. It is entirely possible that they have a whole host of exploits at their disposal similar to the Metasploit Framework. Whichever the method used in this instance, we would expect the FBI to have the budget to not only harness the power of existing exploits but to develop new exploits for undisclosed security flaws.

Of course, developing techniques for de-anonymizing online criminals is not just a task for federal agencies with large budgets. With concerns that the Tor network was being used by pedophiles, H.D. Moore, the creator of the Metasploit Project, created a decloaking engine that aids in the locating of these criminals [6, 7]. He demonstrated these tools through the website `decloak.net`. Through the use of a few scripts and plugins, the decloaking engine was capable of uncovering the real IP addresses of some users hiding behind the Tor network. These tools generally fall into one of two categories. The first category is tools that leak identifying information. One such instance is when the

user uses SOCKSv4 to connect to the Tor network. Because SOCKSv4 does not properly handle DNS resolution, a unique domain name look up can be matched with the DNS server in use by the client. The user can then, possibly, be located through DNS logs. The second category is tools that bypass the proxy altogether. In the case of this decloaking engine, it uses multiple plugins including Flash, the Microsoft Office plugin, and QuickTime just to name a few. These plugins can bypass proxy settings, specified by the user, and make a direct connection to a target computer. A server is set up to accept these connections and the information, including the IP address, is then logged.

As we can see, anonymous networks can be used for both good and bad. While allowing the non-criminal user to remain anonymous on the Internet, it is important to have tools to track down users with criminal intent. The purpose of this thesis is not to present new methods for uncovering users hiding behind these networks, but to give an overview of the existing methods of deanonymization and assess their potential for use in a law enforcement investigations.

Chapter 2

Anonymous Networks

In the following section, we will look at two networks, Tor and I2P, that share the same goal of providing user anonymity on the Internet. They even share a few similar mechanisms for achieving that goal. However, they both differ greatly in their end results. Tor's goal is to provide anonymity for users to access the Internet as it currently exist. I2P's goal is to build an anonymous system on top of the Internet as it exists.

2.1 Tor

Tor is a low-latency, overlay network designed to protect a user's identity when using various applications that communicate through TCP [8, 9]. The prime example is browsing the web. While using Tor, a user can browse the web (or use other TCP-based services) without leaving a trace of his or her IP address in the logs of the various web servers they visit. This is accomplished through the use of a user level application referred to as an Onion Proxy. The Onion Proxy starts a local SOCKS proxy on the user's system which can be connected to by other applications on the system. The data is sent to the SOCKS proxy where the Onion Proxy sends the data through the Tor network.

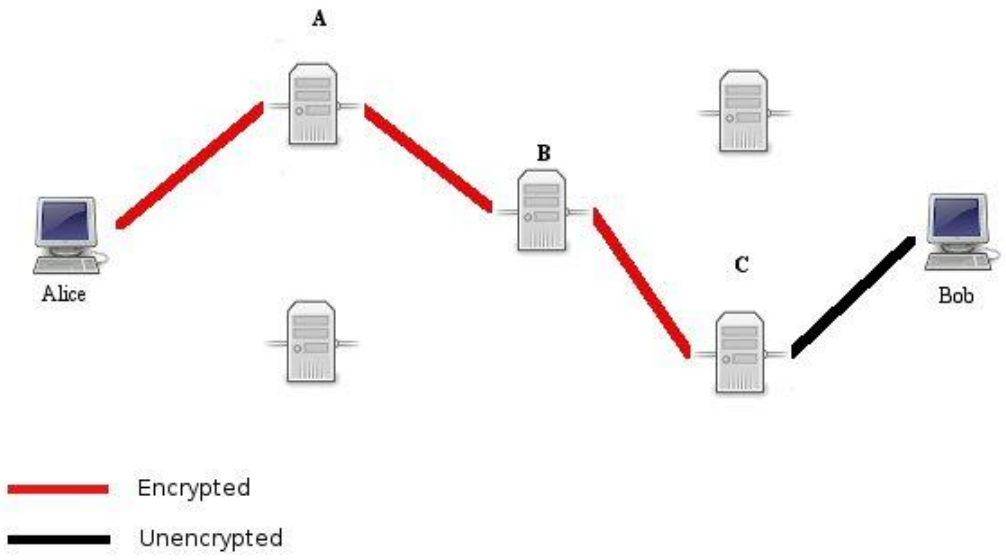

Figure 2.1: A sample Tor circuit between Alice and Bob.

The Tor network consists of volunteer run systems referred to as Onion Routers. The only official systems in the network are that of the Directory Servers. The Directory Server provides a list of all systems running the Onion Router software. Along with the node's IP address is the type of router (i.e. entry, relay, exit, etc.), the exit ports allowed, and the public key of that server. The Onion Proxy starts by connecting to a Directory Server. From here, it collects a list of all nodes and selects a minimum of three at random. After three or more nodes have been selected, the Onion Proxy begins to incrementally build circuits through these Onion Routers. We will refer to both the individual link between nodes and the entire path through all nodes as circuits. The final node (i.e. exit) in the circuit will communicate on behalf of the OP. However, the exit node will not know

what system is running the OP software. Likewise, each node only knows its direct successor and predecessor to prevent identification of the source as seen in Figure 2.1.

It is also important to note that the traffic is encrypted in layers to prevent identification by an attacker running multiple nodes. This is achieved through the use of both asymmetric encryption and symmetric encryption. During the circuit building phase, the source generates a symmetric encryption key for each of the nodes in the path. These keys are securely exchanged using asymmetric encryption. Thus, no node in the circuit knows the symmetric key that the user shares with another system. The message to be sent is then encrypted with each shared-secret key starting with the exit node's key. This provides a message with three layers of encryption. When the message arrives at a node, it knows to decrypt the message and forward it on to the next node until the message reaches its destination. This process can be seen in Figure 2.2.

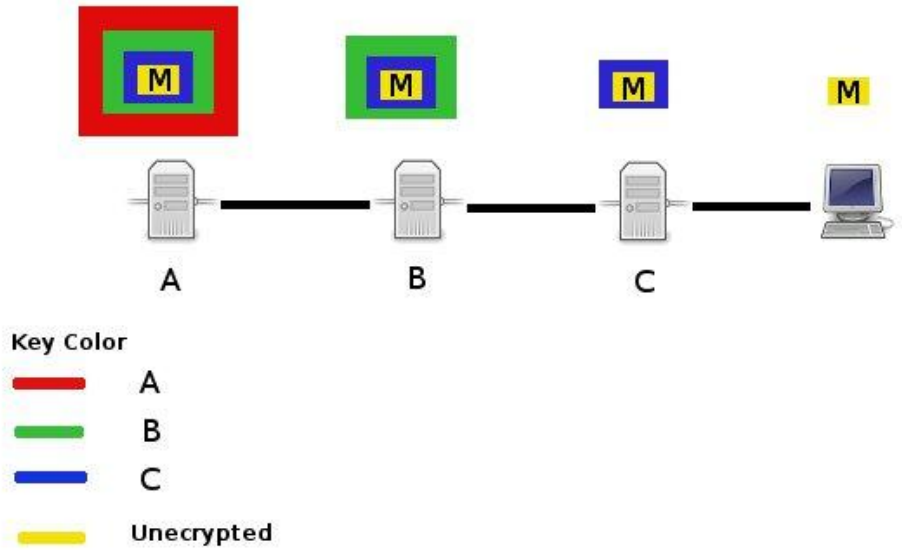

Figure 2.2: Onion routing by Tor.

Any responses sent from the destination are then encrypted in the reverse order. At each stop between the responder and the initial sender, another layer of encryption is added until the message arrives at the initial sender. Because the initial sender knows each key, it can decrypt the message. This method of encryption allows the user to hide either the final destination or the source depending on the nodes location in the path. However, we will see how this method can be attacked in the next section.

2.2 I2P

While I2P [10] shares some similarities with Tor, it is vastly different in its overall goal. I2P is, essentially, its own anonymous "ecosystem" within the Internet. The goal is to protect the identity of the sender and the receiver even from each other. In the Tor network, it is common for the sender to know the location of receiver (unless the receiver is using the hidden services feature). This is not the case with I2P. While it is possible to browse what we can refer to as the standard web, this utilizes a special type of node in the I2P network called an "outproxy". The intended use of I2P is to have users access content provided by other users of the network. Applications are designed specifically for use with the I2P network. In the Tor network, we saw a separation between onion routers and onion proxies (or clients). In the I2P network, all users route traffic for the network.

Inbound and outbound tunnels are key components of the I2P network. These are outbound tunnels and inbound tunnels. If Alice wanted to protect her identity while sending information, she would use an outbound tunnel. Likewise, if Bob wants to protect his identity and still receive information, he would use an inbound tunnel. These tunnels

can be viewed similarly to the circuits used in Tor. Alice's outbound tunnel is going to consist of a minimum of three nodes in the network using layered encryption to ensure each node only knows its predecessor and successor. All communications Alice has with the network will be routed through this outbound tunnel to protect her identity. On the other end of the communication, Bob has his inbound tunnel protecting his identity. Once again, this consist of a minimum of three nodes using layered encryption. Bob will then inform the network that he can be reached through the starting point in that inbound tunnel (also known as the inbound gateway) and this information will be stored in the network database. We can see a simple example of these tunnels being used in Figure 2.3.

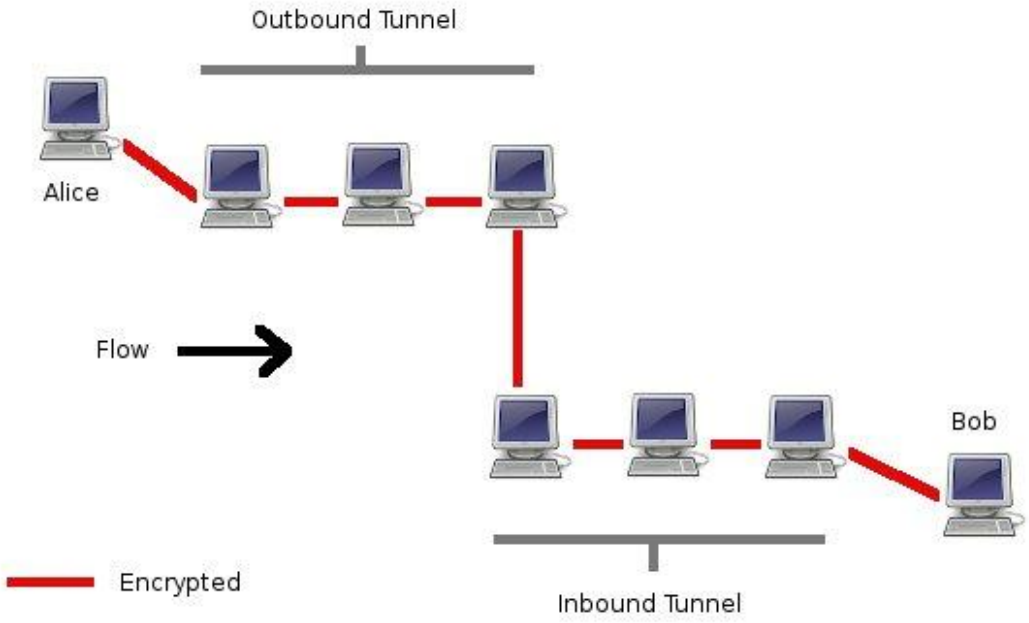

Figure 2.3: Alice and Bob communicating with tunnels.

Now, assume Alice and Bob want to communicate. Alice is going to check the network database to find the location of Bob's inbound gateway. When she has this information, she will use her outbound tunnel to contact Bob's inbound gateway. A key exchange will take place between Bob and Alice using Diffie-Hellman Station-to-Station protocol. This prevents data from traveling between the tunnels unencrypted. Any response from Bob will be sent through his own outbound tunnel to Alice's inbound tunnel. This is the same process in reverse. Alice and Bob can now communicate without either one knowing the other's location.

Chapter 3

Attacks on Anonymous Networks

This section covers attacks on anonymous networks. Each attack is categorized based on the method used. Categorizing attacks on anonymous networks is not easy because many attacks use a combination of other attacks to achieve the end result of identifying the user to the network. Every attempt was made to identify the most important function of the attack and file it accordingly.

3.1 Multiplication Attacks

We will start our section on flow multiplication attacks, simply referred to as a multiplication attack, with a loose definition. A multiplication attack occurs whenever the attacker introduces new packets, or cells in the case of Tor, in the victims path for the purposes of identification by a node later in the flow. This can occur in either direction of the path. What makes each attack unique in this section is the method of introducing new cells into the victim's circuit. One method introduces new cells through an injection

process while another uses Tor's protocol to introduce new cells. In either case, the attacker is required to maintain a presence at more than one location of a three hop circuit.

Fu et. al. Determined the anonymity of a Tor user could be compromised by controlling both the entry node and the exit node of that user's selected path [11] as seen in Figure 3.1. This is accomplished through one of four methods: duplicating an existing cell, modifying a cell, inserting a new cell, or deleting a cell. One of these methods must be used while TCP stream data is being sent over the network. If the attacker attempts to replay, modify, insert, or delete a cell during the initial circuit building phase, the process will fail. The *replay method* works by sending a cell that has previously been sent. By duplicating a cell, the AES counter in the circuit is thrown off. This causes a decryption error which propagates to the exit node. Because the entry node knows the source of the TCP stream and the exit node knows the destination of the stream, the two nodes can work together to de-anonymize the system behind the connection. The *modification of a cell* can work in an attacker's favor in a similar manner. The attacker can modify the first byte of data in the encrypted payload to zero. This will cause a cell recognition error that ends up at the exit node. Even though the packet will fail integrity checking, it can be used to uncover the source. The *faked cell* works much like the modified cell in the sense that the attacker inserts random data into a cell and the error continues to the exit node allowing for identification of the source. The *deleted cell* method works similarly to the replay cell. However, in this case the first relay cell is deleted from the stream while all cells following are sent as if nothing happened. The error is noticed by the relay node and sent on to the exit node where identification of the source can occur.

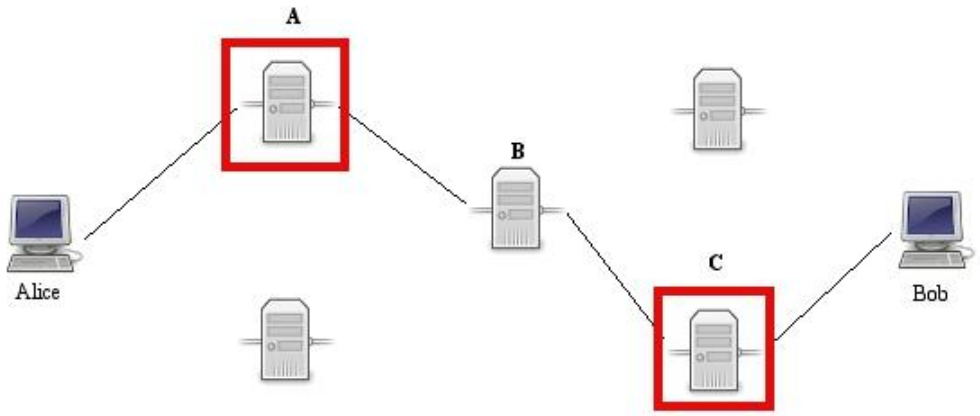

Figure 3.1: The attacker controls the entry and exit node in the circuit.

The developers of Tor consider this a tagging attack discussed in the original design paper [12]. They also stress that this active attack carries the risk of being noticed by the target user. As of this writing, we can find no evidence that this attack would not work against the current implementation of the Tor network. However, this type of attack would be difficult to mount against a targeted system. As mentioned in the paper, it requires the attacker to control both the entry node and the exit node. Furthermore, the user being targeted would need to establish a path using both of the malicious nodes.

Ling et. al. devised a method of encoding using the cells in the Tor network to compromise a user's identity [13]. As with other similar attacks [11], the attacker controls the exit and entry nodes in the path. The attack starts when the exit node receives return

data from the end point in communication. The node will then log the IP address and port number of this endpoint and link it to this circuit. This is where the encoding mechanism is necessary. The cells are stored in a circuit queue until the node flushes the data into the network output buffer. When a write command is called, the cells are sent to the next node in the circuit. In the case of a three-hop circuit, the data is sent to a relay node. As we know, the relay node is the only node in the path that is not controlled by the attacker. By modifying the number of cells flushed to the network, the attacker can encode a signal that can be recognized by the accomplice entry node. Flushing three cells would indicate an encoded "1" bit, and flushing one cell would encode a "0" bit. Meanwhile, at the entry node, it has logged the start point of the stream to compare the encoded stream with the exit node. Thus, by modifying the exit node to store cells until the proper encoding was achieved and then modifying the entry node to decode the signal, the attacker could identify all parties in the identified stream.

There is no indication that this attack will not work with the current version of Tor. The authors cite papers stating an attacker would require control of only ten percent of the onion routers in the Tor network to confirm half of the communications. As Tor grows with more people offering their systems as onion routers, that ten percent becomes larger and larger. Even if the attacker were to use a resource such as PlanetLab, launching a targeted attack would be difficult. A user might become suspicious when they see a list of nodes all hosted by such a source. Because this attack requires control of both the entry node and the exit node, it is unlikely to be useful outside of the academic world or in a real world untargeted attack.

Wang et. al. Proposed a flow multiplication attack against Tor [14]. This is, essentially, a mix between an injection attack and a pattern matching attack. It is assumed that the attacker has control of both the entry node and the exit node on the path between Alice and Bob. A normal path through the Tor network is established by Alice. When Alice requests a web page from any web server, the malicious exit router recognizes this from the relay cell that is sent. Instead of immediately fulfilling that request, the exit node returns an HTML page containing multiple image source tags. When Alice's web browser receives this page, it must request those images from the source. It is important to note that these images are blank as is the html page that has been injected. Thus, Alice will only notice a blank page displayed by her browser. In the paper, the authors find that an image request of this nature will generate predictable patterns in the circuit. Specifically, Alice will generate two upstream relay cells and three downstream relay cells. This can be observed at the entry router. The malicious entry and exit node are synchronized using Network Time Protocol. Once the injection is made at the exit node, Bob's information is logged because this was the original destination of the request. The entry router has also logged Alice's information. It, then, waits for the pattern of the requested images. When the entry node has received the predicted pattern, the two nodes can then link Alice and Bob to that specific stream. Thus, Alice's identity has been compromised.

The difficulty with this attack is the requirement of controlling both the entry node and the exit node in the Alice's path to Bob. This makes de-anonymizing untargeted users easy, but not necessarily a specific targeted user. The authors offer a suggestion of exaggerating the actual bandwidth of the malicious routers to further tempt Alice to use the attacker's nodes, but this comes with its own side effects. Not only will this tempt

Alice, but it will tempt other users. Because the node has suggested it has more bandwidth than it can actually provide, it might actually take more connections than it can handle. As for the whether or not this attack will work with the current version of Tor, we cannot find any information indicating it would not work as advertised.

3.2 Intersection Attacks

The goal of an intersection attack, to put it simply, is to narrow down the possibilities. As it is discussed in [15], the attacker wants to reduce the victim's anonymity group. In the case of the Tor network, this could mean reducing the number of nodes the victim potentially uses, or it could mean looking at user's online and offline behavior.

Imagine Mallory wants to figure out who is communicating with Bob anonymously. She can observe Bob's traffic and know what times he is being contacted. She knows the communications are coming from a specific anonymizing network. Mallory can then look at the set of users accessing this network during Bob's online times and come up with a subset of suspects. From this subset, Mallory can further reduce the possibilities through continued observation of users accessing the network during Bob's online time. We see the subset reduction from one communication period to another in Figure 3.2. Mallory eventually reduces this set to a single element of Alice. As mentioned, this is just a simple example. It would only be practical on a network with a single ingress point that does not adjust timing.

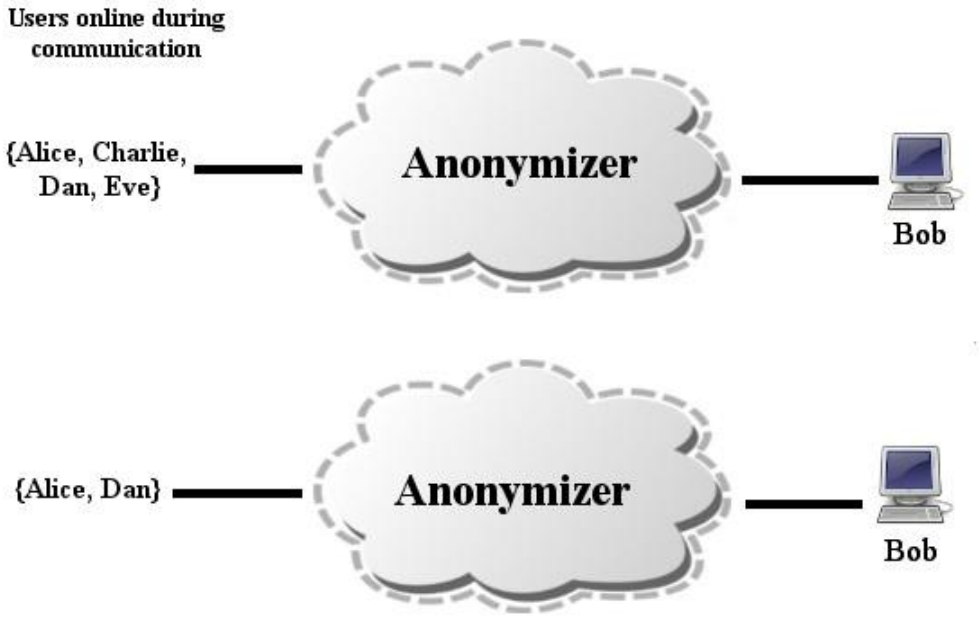

Figure 3.2: The resulting subset after two observations.

Berthold et. al. lays a foundation for intersection attacks that was built upon by other authors [15]. The authors describe an intersection attack while attempting to build a model for the perfect anonymizing system. It is comparable to profiling a user of the system. The authors state a user generally follows certain patterns in web browsing. This includes sites visited, online and offline times, email services used, etc. By building a set of all users on the system, the attacker can use an intersection of these elements and active users to narrow down an initiator of the communications.

While Tor was not the anonymous communications technology focused on in this paper, it is important to look at how this type of attack might affect a user of the Tor network. This type of attack is addressed in the Tor design [9]. One of the advantages of

the Tor network is that the user is essentially hiding among the crowd. The larger the network becomes, the more difficult it is to profile a single user. The fact that a user's path is rebuilt approximately every ten minutes makes an intersection attack even more difficult. Assuming the attacker is watching from a given exit node, they are only given about ten minutes to profile that user before a new circuit is built and the user likely has a new exit node.

Back et. al. focuses on the freedom network and PipeNet [16]. However, the authors indicate that these attacks can apply to other anonymizing networks. The authors look at four generic attacks that are a potential threat against anonymous networks. This includes: packet counting attacks, attacks on traffic shaping, latency attacks, and *clogging attacks*. In the standard scenario involving Alice communicating with Bob, the *packet counting* attack attempts to link the two together. The idea is to sniff packets between Alice and the first node in her path. The author suggests the use of Alice's ISP or a router on the path between Alice and the first node. The packets exiting the first node are then counted to correspond with the packets received. This is done for each node until Alice's communication can be linked to Bob. The second attack presented is on *traffic shaping*. This method is based upon systems that present the user with a limited bandwidth upon connection. The attacker must guess the path chosen by Alice using whatever means are available. The circuit is chosen and the attacker then sends traffic until the bandwidth limit is reached. Nodes will then cease to send traffic between each other and the real throughput can be seen. The third attack is a basic *latency attack*. The attacker computes the latency of multiple paths within the network. Statistical methods are used to account for noise in the network. This will account for any random delays normal Internet traffic

might encounter. The attacker then must estimate or calculate the latency between Alice and the first node (calculate if the attacker controls the first node). By using these latency sets, the attacker can determine that Alice is connected to Bob with some certainty. The circuit clogging attack presented in this paper is discussed in Section 3.4.

Li. et. al. identify, what they call, "super nodes" in the Tor network. These are nodes with high availability and high bandwidth. Essentially, the nodes are the core of the Tor network. This is important because this core consists of 21 % of the network, but they carry approximately 66 % of the traffic. These nodes are also significant because the path selection algorithm is biased towards these nodes. While the authors classify the attack presented in this paper as *loop attack*, we will classify this attack as an intersection attack because the attacker must create a subset of nodes from the network in which they will focus on.

The attack begins by identifying the super node subset of the Tor network. With this knowledge, the attacker begins to take nodes from this subset offline (ISP level or denial of service). After continuous downtime in the network, users begin to lose confidence in the network. After each cycle, the number of users of the network drops due to the inability to access the network. This, in turn, reduces the remaining users' anonymity set. One of the cornerstones of Tor's anonymity model is that users are "hiding among the crowd". If there is a small crowd to hide among, the anonymity level is reduced and the attacker can begin to compromise identities through other attacks.

This type of attack is more likely seen from at the ISP or Government level. It is certainly not outside the realm of possibility for a single attacker if they were to gain

knowledge of a vulnerability in the Tor code. A denial of service attack could be launched to bring down super nodes in the network and significantly reduce the performance of the network. However, such an attack would be highly visible and, hopefully, quickly corrected. Because the attack presented in [17] did not bring down the entire core to affect performance, an ISP providing service to a significant number of super nodes (2% of regular nodes and 10% of super nodes in the paper) could have a great impact on the performance of the network. Of course, this remains a highly visible attack.

Wright et. al. focus on intersection attacks against peer to peer anonymity networks [18]. They note that the Tarzan and Crowds networks are especially vulnerable to their intersection attack. The observation period is considered a round. For each round the responder is contacted by the initiator, a set of users accessing the network during that time is built. The attacker can then take the intersection of the previous round's set to produce a subset. This subset continues to shrink until the attacker has either one suspect or few enough suspects to launch another attack on. The authors were able to show that an intersection attack does not necessarily have to be performed by a patient attacker. However, this was before the Tor network. Tor has multiple ingress and egress points that make it difficult to monitor. Thus, the intersection attack based on online and offline times of communication is much harder to perform successfully. However, there is an exception to this when the attacker suspects to people communicating. In this case, the attacker can use online and offline times, but even this is not guaranteed.

3.3 Fingerprinting Attacks

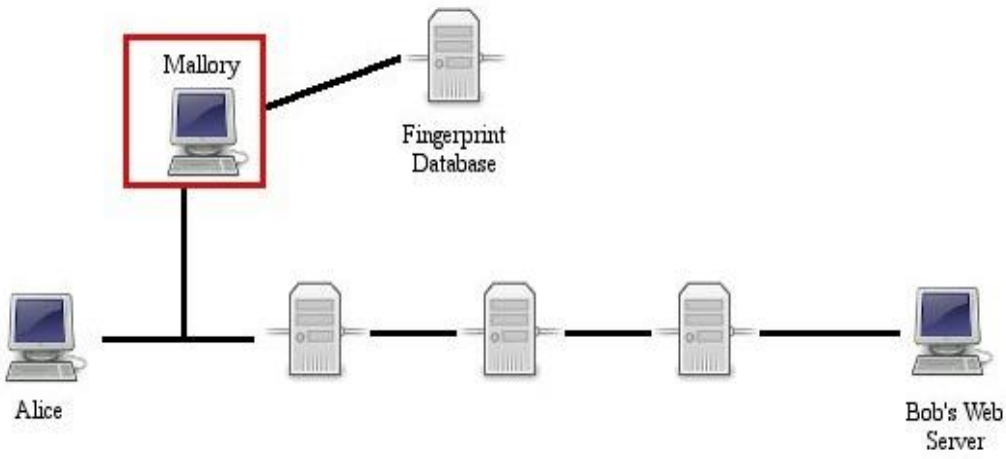

Figure 3.3: Mallory comparing Alice's traffic to here database.

A fingerprinting attack works much like a detective checking fingerprints from a crime scene. The detective preserves the fingerprints for comparison against a database of previously collected fingerprints with the hope of finding a match. However, in the case of this attack, the victim's traffic is being collected for comparison against a previously collected database of traffic signatures. Even encrypted web traffic, while not decipherable without the proper key, can still have a signature to it. For example, if Alice visits `randomwebsite.com` while using the Tor network, it will require the same number of cells for each visit assuming the site has not changed. If our attacker, Mallory, is watching Alice's traffic between Alice and her first node, Mallory can see this signature. If Mallory's database of know signatures contains the trace for *randomsite.com*, she can link Alice to that site even though she cannot view the actual details of the traffic. Of course, this becomes a problem with dynamic sites, but we will discuss that later. Figure

3.3 shows Mallory monitoring Alice's traffic and comparing it to her database to determine the destinations.

In order to pull off an attack on a multi-hop network consisting of a compromised entry node and exit node, the user must select both nodes in the attackers chosen path. The attacker can set up the malicious nodes and just hope the victim will utilize them when selecting a path or the attacker can exploit the path building process to increase the likelihood the victim selects the malicious nodes. Bauer et. al. propose just such an attack [19]. Their *low-resource routing* attack uses Tor's node selection algorithms against the network to further increase the chances the victim will select the compromised nodes. Once the victim has selected the attacker's entry node and exit node, many attacks in this paper become possible. There are two node selection algorithms at work in the Tor network. The first algorithm is responsible for selecting an entry node for the user's path. It uses two criteria when selecting a suitable node. The first is the nodes up-time; nodes that have an up-time greater than the median up-time for the other nodes in the network are the preferred. The second is the resources available from a given node; the nodes that have more bandwidth to provide will have a better chance of being selected as the entry node. However, the authors have discovered that the trusted Directory Servers do not verify the resources reported by a node. The node simply reports to the Directory Server while the Directory Server accepts that report. By reporting inflated resources, the attacker can greatly increase the chances of having the victim select the malicious node. In a test network of 60 standard nodes and 6 malicious nodes, the authors were able to control more than 46 % of the traffic in the network. Tor then uses a differing method of selecting non-entry nodes. To ensure the majority of nodes are utilized in the network,

less importance is place on up-time and resources. Simply allowing all ports to be used on exit can increase the attacker's chances of becoming the exit node in the path. As for the specific fingerprinting technique discussed, the authors state that only the entry node needs to be compromised in order to perform this attack. They suggest an attacker may coerce the user into visiting a predetermined site. The attacker will collect packets which, even though encrypted, build a watermark for a user visiting that particular site. When the user visits this site, the attacker can match that packet pattern to the previously built pattern. The attacker has then confirmed the sender and receiver in that particular stream.

As of the time of this writing, we can find no information indicating that low resource routing attacks will not still work against the Tor network. As mentioned previously, the larger the Tor network grows, the success of this type of attack reduces. A greater number of compromised nodes will be required by the attacker to ensure the victim is using those nodes.

Herrmann et. al. present a generic fingerprinting attack for use against tunneling methods [20]. These methods can include OpenSSH, Stunnel, Tor, etc. There are two phases to the attack. In the training phase, the attacker collects fingerprints for each website the user could possibly visit and stores it in a database. The assumptions taken for this portion are that the attacker knows the method utilized by the victim and the web pages that the victim visits. So, before the victim starts browsing, the attacker will use the same tunneling method as the victim to visit each site possible. From this training phase, the attacker will build a database of fingerprints from the encrypted traffic. The second phase is the testing phase. The assumption here is that the attacker has the ability to

observe the traffic from within the tunnel. In other words, the attacker is sniffing traffic between the victim and the tunnel end-point. The attacker then compares this traffic to the fingerprints taken in the training phase to match the victim's communication with a website in the list.

The author's results are very promising when it comes to single-hop tunneling methods such as OpenSSH and Stunnel. They achieve an accuracy level in the low to high 90s. However, it is important to remember the assumption that the attacker knows all possible websites the victim could visit. In a real world application such as a police investigation, this assumption is unrealistic. A simple single term Google search could provide far too many results for a fingerprint database to hold in a realistic scenario. While the single-hop system testing provided a high accuracy rating, multi-hop systems such as Tor and JonDonym (formerly JAP) showed this method to be unusable. With accuracy below 3% when attempting to identify Tor traffic, this method is not usable without modification.

Shi et. al. looks at the original fingerprinting attacks that were leveled against networks such as SafeWeb [21]. When the user of the network visited a specific web page, it would require the user to download additional objects within this web page such as images and music. In a standard browser, each object would initiate a separate TCP connection to download that object. These files could then be observed based on file size despite being encrypted. The user could then be identified by an attacker observing the communication. However, because Tor uses 512 byte cells, this attack is not possible by simply observing the traffic created by these objects. Tor also uses the same stream to obtain these objects and thwart the potential attack. The authors offer a modified version of this attack. They

realize that each object uses a numbers of cells in what they call an interval (e.g. the inbound traffic for an object without the outbound traffic). They then define a vector that consists of these intervals. Another vector is constructed to match a specific webpage. Unlike some other attacks, this method requires the attacker only to control the users' entry node in the Tor path. As the user browses the web, the malicious entry node observes the traffic. While the user is browsing, the interval vector is being created. Portions of this vector (we assume the user to visit multiple pages) are then compared to the previously built fingerprint vector. If the two vectors match, the attacker can assume the user is connected to the webpage built for that vector.

The authors admit that building a fingerprint for a dynamic web page is difficult. They cite `youtube.com` and `amazon.com` as prime examples. Content on the sites are constantly changing and in some cases, the content is unique for each user. A very large database of fingerprints would be required to guess the users' destination with any type of certainty. We also suspect this attack would suffer from many false positives. However, this attack method does require far fewer resources in the sense that only the entry node is required to mount the attack. If the attacker were able to guide the user somehow to a web page uniquely crafted for this purpose, the attacker might be able to decloak the user, but this would require the user to select the attacker's malicious entry node in the victim's path.

3.4 Congestion Attacks

Circuit clogging, or congestion attacks, tend to use a system or networks resource monitoring capabilities against themselves. While these metrics can be useful in helping the network run at peak performance, they can also be detrimental to the anonymity of the users. We will see a few variations on this attack in the following, but the general premise for the attack is to modify traffic in such a way that the victim's path through the anonymous network becomes visible to the attacker. Once the path is known, anonymity for the victim is broken.

Back et. al. laid the framework for a clogging attack against a generic anonymity network [16]. The attacker sniffs the traffic between an exit node and the final destination. While watching this connection, the attacker creates a path and uses as much bandwidth as possible. If the flow between the exit node and the end point decreases drastically, the attacker knows that node is on the chosen path.

Murdoch et. al. put the circuit clogging attack to work against the Tor network [22]. Their goal was to present an attack that fell within the threat model set forth by the designers of Tor. However, they argue that this attack could affect any low latency anonymity system. The attack revolves around the observation that a single node in a path, which is under heavy load, will increase latency for the entire path. In other words, if one of the nodes on the users' path is handling a high volume of traffic, the user will notice longer wait times in their communication through the Tor network. In the scenario presented, the attacker controls a malicious Tor node, n1. This does not have to be a node on the victims selected path. This node, n1, then makes a connection to another node, n2,

in the network for the purposes of measuring the traffic load on the selected node n2. The attacker in this scenario also controls a malicious server, s1, in which the victim is communicating. This server will send specific burst traffic patterns through the Tor network to the victim. Meanwhile, the attacker's Tor node, n1, is monitoring the traffic load on the selected node, n2, for this specific pattern. If the pattern matches, the attacker can make the reasonable assumption that the monitored node is on the path to the malicious server. This can be performed on each node until the entire path is discovered. The author then equates the user's anonymity level to that of a chain of proxies.

It is important to note that the experiment in this paper was performed on Tor version 0.0.9. At this point in time, the network was still in its infancy. It consisted of only 13 nodes. Hence, while the resource requirements to implement this attack at the time were reasonable, Tor has since grown significantly. Evans et. al. argue that this attack is not feasible with the current size of Tor [23]. The bandwidth requirements to test the traffic load on each individual node in the network are beyond the reach of the average attacker. They also note that false positives are very likely when other users increase the load on the network. It is also important to remember that the victims' path will change every 10 minutes. This is problematic for an attacker because the bursts of traffic discussed in this paper consists of a few seconds worth of traffic. The attacker would need to increase the number of malicious nodes and decrease the recognizable burst time to cover every node in the network.

The attack presented in [24] does not necessarily fit into category of circuit clogging attacks, but is very similar. The attacker makes adjustments in the traffic that can be

observed from the outside. While the attacker is not exactly clogging the circuit, they are watching for changes they introduced at each node to determine its status in a given circuit. In this scenario, we assume that Mallory is the attacker looking to locate Alice while she is hiding behind the Tor network. To perform this attack, Mallory will act as the final point in Alice's communication path and he will have another system outside of the Tor network. When Alice connects to Mallory and downloads a file, Mallory will begin to fluctuate the traffic. Mallory's second system will then begin probing node's bandwidth in the Tor network. She can check bandwidth metrics such as capacity and available bandwidth. Because Mallory is fluctuating the traffic, she is able to detect these changes and identify that node as a part of Alice's circuit. Through this method, Mallory is able to determine all three nodes in Alice's circuit and then identify her location.

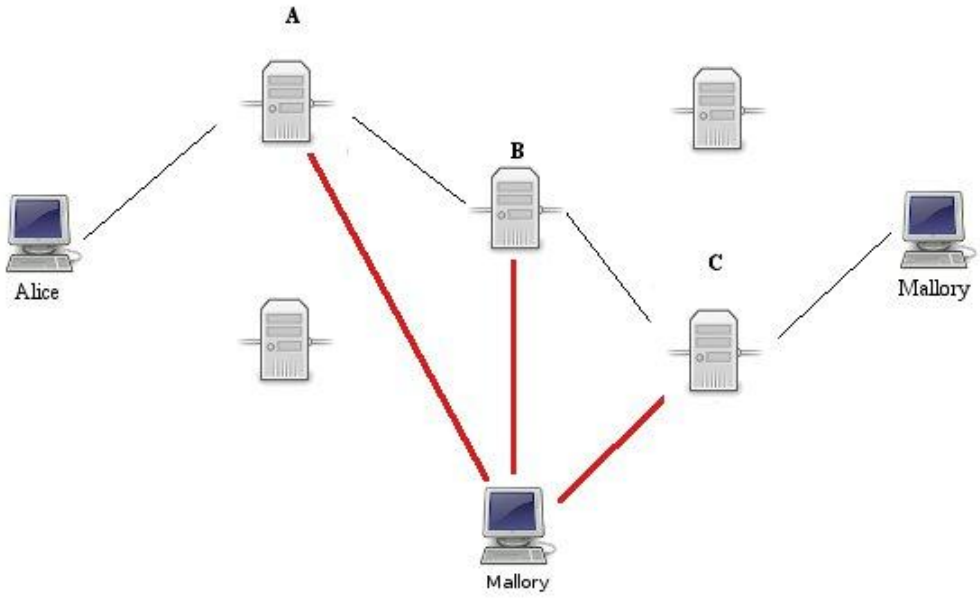

Figure 3.4: Mallory probes nodes while she fluctuates the traffic.

The resource requirements for a single attacker in this scenario is reasonable. The attacker simply has to coax the victim into communicating with the colluding server. However, the problem arises in the reliability of this attack. In the experiment, 11 of 50 circuits were identified with a false positive rate of approximately 10%. The trade off comes in the form of the low resource requirement. Because the attacker is not required to control any nodes in the network, we might be able to accept and identification rate of 22% of all Tor connections to the server. Perhaps, with fine tuning, this method could become more reliable and become a significant threat to the Tor network. We can see an example of this attack in Figure 3.4. Mallory fluctuates traffic over Alice's path while she takes measurements to determine all nodes in Alice's Path

Evans et. al. modifies the clogging attack presented in [22] to attack a larger and growing Tor network [23]. In [22], the attacker was not required to control the victims exit node to perform the attack. In this case, the attacker must have control of the exit node selected by the victim. Other requirements of the attacker are a malicious client accessing the Tor network and a malicious server. The attack starts by injecting JavaScript into a HTML response from the web page the victim is accessing. This script will cause the client to make an HTTP request every second. Each of these requests is handled by the malicious exit node and a blank response is replied. This causes the browser to dismiss it. Local system time is also included by the JavaScript code to account for the scripts potential inability to precisely send requests every second. Before any of this is done, the attacker must establish a baseline so a comparison can be made during the congestion. Thus, the need for a malicious Tor client must be met. A baseline

must also be taken after the attack to ensure the congestion was not caused by overloaded nodes. To perform the actual congestion attack, the attacker must create a path long enough to loop back to a node already selected in the path. This requires the attacker to use a path longer than the default size of three. This is necessary because a Tor circuit will be torn down during construction if a repeat node is detected. The authors suggest a path size of 24 hops. This will cause significant congestion on the selected node. The delay is then compared to the baselines taken before and after. If the delay does not match that of the baselines, the attacker can assume that node to be the victim's entry node. As the exit node and the relay node are already known to the attacker, the attacker obtains the victim's entire path through the Tor network.

One of the keys to making this attack work is knowing every node in the network. In order to overload the node, the attacker must know that it exists. The Tor directory servers provide this information freely. However, the authors point to a newly introduced idea of a bridge (at the time of their writing this was new). The bridge was introduced to help thwart filtering methods used by ISPs. Bridges are essentially unpublished nodes in the Tor network. If the ISP does not have them listed, then the ISP cannot filter them. This also helps block a circuit clogging attack as the one mentioned previously. If the attacker does not know they exist, then the attacker cannot congest them. At the time of this writing we cannot find any information to show how widespread the bridge use is.

3.5 Timing Attacks

Instead of a standard anonymous network, Senger et. al. focused on uncovering the end communication point of a targeted user in the Skype peer-to-peer network [25]. In this case, the authors use assume a suspect under investigation by law enforcement. Assuming the agency does not have access to the suspects' computer, they want to know who the target individual is communicating with. Skype presents two challenges to this problem. First, all traffic is encrypted using 256 bit AES encryption. Second, Skype is a peer-to-peer overlay network with multiple legs of communication to route the traffic to and from its destination. The authors propose a method by which patterns are embedded in the traffic that is recognizable to the monitoring entity. In this case, packet departure times are manipulated by nodes designated as Call Markers. Each call marker, deployed at the various edge routers, receives a specific signal function from the entity referred to as the Call Tracker. This provides what is, essentially, an ID for that Call Marker. Then, through the use of a Discrete Fourier Transform, the adversary can identify the pattern and know which edge router the traffic originated from. Thus, the end point in the identification can be discovered.

While the technique is sound, the requirements are great. Many anonymizing technologies assume that the attacker does not have global capabilities. The attack requires cooperation from the ISPs involved in the attack. The goal of the attack is to identify the end point of communication for a targeted user. Thus, every possible ISP would need to cooperate with the law enforcement agency. Even though every router in

the path is not required to identify the two suspects communicating, we must still assume a global attacker in this case.

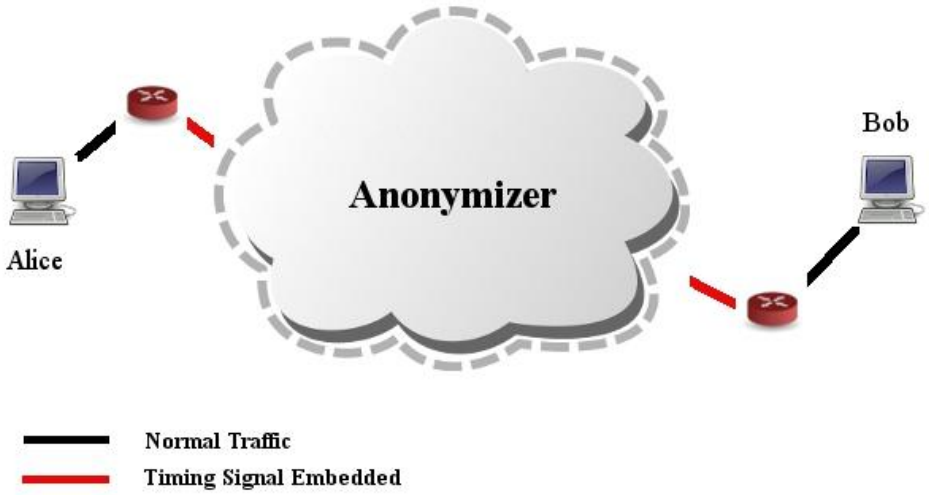

Figure 3.5: The timing signal is inserted and removed from the circuit.

Moreover, Wang et. al. present another timing attack focused on the Skype network [26]. However, this time the victim is using an anonymous network to hide the source and destination of the communication. In this particular case the anonymous network is *findnot.com*. The purpose of the attack is to determine if user A and user B are communicating using Skype with the constraint that the attacker does not have global capabilities. In other words, Mallory suspects that Alice and Bob are communicating, but cannot confirm this because one or both are using an anonymous network to mask the origin of communication. Through the inter-packet timing of select packets, the authors are able to apply a watermark to the packet flow that allows them to link the

communication between Alice and Bob without disrupting the real time service provided through Skype as seen in Figure 3.5.

As with the previous method discussed in this section, this attack appears reliable. However, the main question raised is about its usefulness. A considerable requirement is the attacker should know about both Bob and Alice. The attack can then link a communication between the two using the Skype network. A major problem is that the communication is still encrypted. The law enforcement agency is still unaware of any criminal activity unless, of course, the act of Alice communicating with Bob is the criminal activity.

A similar attack is presented in [27]. The attack works on the same principle, but it uses an invisible traceback. In this attack, the adversary suspects Alice and Bob are communicating. However, the adversary cannot be certain because Alice and Bob are using an anonymous network to hide their locations. The authors rely on two main components to determine if Alice and Bob are, in fact, communicating. These two components are the mark generator and the mark recognizer. The mark generator inserts a signal in the communication that can be recognized on the other end of the communication by the mark recognizer. The recognizable signal is inserted through the use of a chipping code. The authors refer to this as a pseudo noise code. The recognition system sees this signal inserted into the stream and Alice and Bob are now known to be communicating.

With this attack, we run into some of the same issues as the previous methods. The attack will work if the attacker knows both users in the communication, which is not always the case. Sometimes only one entity is known in the communication and the

attacker is attempting to uncover the other party. For example, if Alice is visiting a website that belongs to the attacker and she is using Tor to hide her location, the attacker does not actually know who Alice is and this attack becomes less effective. The attacker must have some kind of hunch of where Alice is so the recognition portion of the attack can be properly placed. Even if we know the two parties communicating and can verify this fact, anonymizing networks generally employ some kind of encryption to hide the payload from an outside party. Thus, this type of attack is only useful in the event that both parties are known and we need to verify the communication between them.

3.6 Application-Level Attacks

Attacks at the application level are generally not a sign of weakness in the anonymous network, but they occur through the use of an application that does not take privacy into consideration during implementation. The goal of an application level attack can be placed into one of two categories. Both methods attempt to gain the targets real IP address. The only difference is the avenue of delivery for that IP address. The first method is to send the IP address or identifying information through the anonymizing network within packets. In other words, the information still travels through the anonymous network, but it is encapsulated in the data being sent. This is possible because anonymous networks, such as Tor, do not filter the actual data being sent [9]. The second approach is an attempt to get the target application to establish a connection outside of the anonymizing network. This involves the application bypassing the local proxy and connecting directly to a server waiting to log the information.

The first attack uses the *information leak* method in which the user's IP address is actually sent through the anonymizing network as seen in Figure 3.6. In classic Alice and Bob scenario, the application Alice is using has just sent a packet containing her IP address and compromising her identity. An example of this method is seen in the use of torrent applications over the Tor network. In a recent paper [28], it was discovered that some Bit torrent applications willingly reveal a user's identity through the initial communication with the torrent tracker. This occurred despite specifying the use of a SOCKS proxy within the application.

Figure 3.6: Alice's IP address is sent in a packet through the network.

When a user joins a torrent to download a file, the application must announce itself to the tracker specified in the torrent file (unless a distributed hash table is being used). This announcement is done in the form of an HTTP get request. This request contains the

necessary information to join the torrent and alert the tracker that the client is either looking to share the file or download the file. However, in the torrent specification, the request contains the source IP address as an optional field. Thus, depending on the application, the user could be sending his or her real IP address directly to the torrent tracker. Of course, uncovering the user depends on a number of things. The first is how the system connects to the Internet. If the client system is on a private network behind a NAT, the private address will be sent to the tracker. However, if the client system is connected directly to the Internet, his or her actual public IP address will be sent to the tracker. The second dependency is the torrent application in use. Because Bit torrent is an open specification, anyone is free to develop compatible applications. The authors of this paper [28] found multiple applications that were leaking identifying information. However, from our own testing, we have found that many of the applications listed in the paper have since dropped the optional field from the communication.

The authors were able to further expand their Bittorrent attacks in [29]. Because Tor uses a single circuit to transmit data over multiple streams, the information gained in previous attacks was used to link specific circuits in a malicious exit node to an IP address. For example, the IP address of a user is uncovered during Bittorrent communication. Any further traffic coming from this circuit can be identified as coming from the same IP address. Bittorrent also allows the victim to be tracked across other exit nodes controlled by the attacker because of the 20-byte peer identifier used in the communication.

The second attack method was explored by H.D. Moore, the creator of Metasploit. He developed multiple tools to compromise privacy and deployed them at the website `decloak.net` [6]. All of these methods use the second avenue of delivering identifying information. `decloak.net` attempt to bypass proxy settings on the client system. It accomplishes this through the use of DNS, Java applets, Flash, or other third party plugins as shown in Figure 3.7. In this case, Mallory has forced an application on Alice's computer to establish a direct connection to his logging server after the initial secure connection through the anonymizing network. Alice's IP address can be logged and her privacy is compromised.

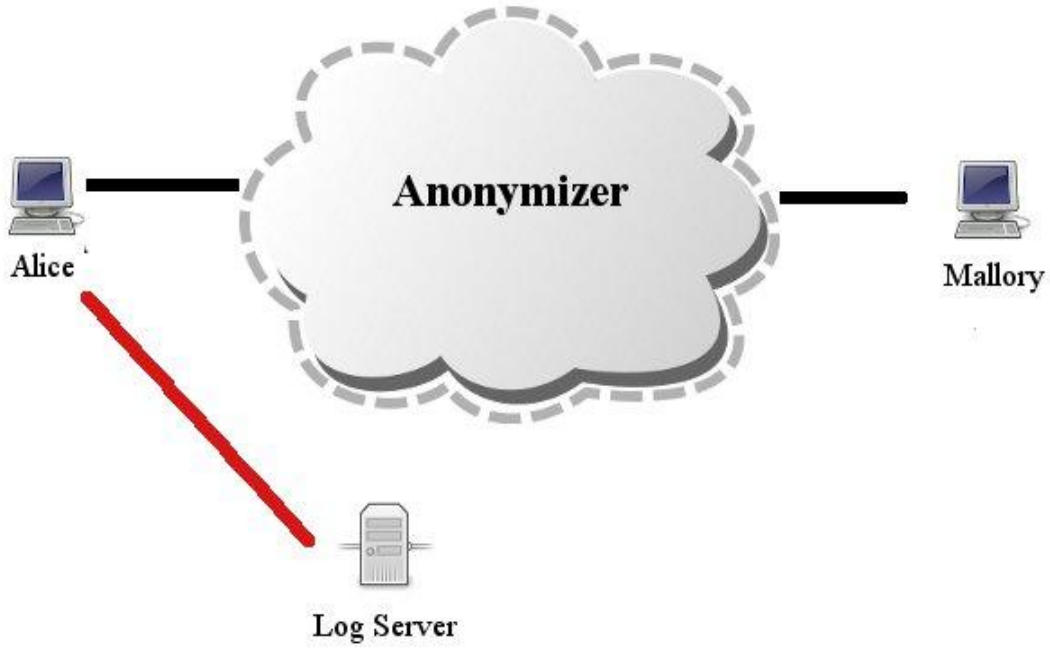

Figure 3.7: Alice is forced to make connection outside of the network.

The DNS method of attack relies on the SOCKSv4 protocol's inability to properly handle a DNS look up. The web page provides a link that consists of a unique identification number appended to the domain name in use. Because the overall domain name will be unique for each individual user, these forces a DNS look up from the client. As indicated earlier, SOCKSv4 does not properly handle DNS look up. Thus, the client application will bypass the proxy settings and perform the look up using the client systems specified DNS server. To identify targeted Tor users, a special DNS server for the domain is set up to log all incoming requests, and the clients DNS server is collected for further investigation.

The Java applet approach consists of two tools that use the design of the Java API against the client. The first part of the tool uses a function to resolve host names in the Java API. If the host name in the function call does not match that of the web site providing the applet, a security exception is raised. However, this does not deter the client system from performing the look up. The proxy is bypassed in this process and identifying information is leaked.

The second method used in the applet based attack involves UDP packets. If a UDP packet is sent using a Java applet, the proxy specified by the web browser is bypassed. Thus, a UDP packet is sent directly to the destination. Once again, a server is set up to collect this packet which contains a unique identifier. This can result in the client system revealing its public IP address.

The final approach to the Java applet tool is rather direct. The Java API actually allows the applet to look up the host name and IP address of the client. This can be compared to

executing the `ifconfig` (or `ipconfig.exe` for Windows users) command on the client system. Because the function call is executed on the clients system, it will reveal the network settings of that machine regardless of proxy settings. However, it is important to note the Tor web site recommends disabling Java for security reasons.

The final portion of the decloaking engine consists of various plugins and applications that can be used to compromise a user's privacy when browsing with Tor. Each method, as previously mentioned attempts to bypass the proxy settings of the client system.

Both the Flash and the QuickTime plugins can be utilized to establish a direct connection to a server. This, of course would bypass the local proxy settings allowing the attacker to log the victim's real IP address. The main difference between the two plugins is how they achieve this result. Flash contains a standard socket function in its API that establishes this connection. We were able to successfully utilize this method over the Tor network. A Flash script was built to establish a TCP connection to our server. If the user is not blocking active content using the Tor Browser Bundle, the browser will hand the script off to the Flash executable. The Flash executable will ignore any proxy settings set in the browser and establish a separate TCP connection directly from the user's computer to the specified server. However, the Tor Browser Bundle blocks such embedded objects from being displayed making this attack ineffective when proper measures are taken to protect privacy. Meanwhile, QuickTime allows the setting of a parameter to establish a direct connection for the purpose of watching a video. The result is the same for both methods if the Tor Browser Bundle is not set up properly, the user's privacy is compromised.

The last two components of the decloaking engine require the existence of certain applications on the target system. The first application is Microsoft Office. When Office is installed on a system, a browser may choose to allow the viewing of documents within it. When the browser is configured to automatically open documents the file may require the downloading of an image to complete. This will bypass the local proxy settings and allow a direct connection to the target server. The real IP address of the client system can then be logged by the server and the user's privacy compromised. Moreover, this mechanism will also work when the user opens the document at a later time. In this case, each targeted anonymous user should be provided a file with a unique image URL.

Similarly, if iTunes is installed on the target system, the web server may provide a link to a page containing `itms://` in the URL. The protocol handler will then open the iTunes application using the link specified. This will establish a direct connection to the specified domain where the attacker can log the user's real IP address.

Chapter 4

Conclusion

We have looked at multiple methods that have been successful at de-anonymizing users at one point or another. The goal of this thesis was to compile all of these attacks and assess them based on what a state or local law enforcement investigator might be capable of. The issue arises in the fact that the investigator might not only be limited in the resources available to him or her, but the law also limits what an investigator can and cannot do. For example, let's say the investigator implements an attack that requires the control of the entry node and exit node in order to catch the bad guy. Even if, the attack targets the bad guy only, he or she must still monitor traffic on that exit node. This includes users who have nothing to do with the case. Of course, the laws vary greatly from country to country, but, generally, a warrant is required for such monitoring. The equivalent in wire-tapping can be seen as listening to many phone calls from multiple people until the officer finds the correct one.

The requirements for de-anonymization attacks vary greatly. Some require resources that could be out of reach of an investigator while others could require as little as a

computer running a web server. For example, an attack that requires control of both the first and last node in a circuit is difficult to implement against a targeted user. If the goal is de-anonymization of many unspecified users, this approach is reasonable. However, getting a specific user to use both of the rogue nodes could prove to be very difficult.

It is important to remember that much like the Tor network, the information in this thesis can be used for both good and bad. We have covered methods that can be useful for law enforcement attempting to track down a known cyber criminal that hides behind anonymizer technologies. On the other hand, these methods could be used by an oppressive dictator's intelligence service to further crackdown on dissidents within their country. Of course, this thesis is also useful to the user's of anonymous networks. For example, a subject of the previously mentioned oppressive regime could use this information to better protect his or her communications that might bring unjust punishment. Technology itself is neither good nor evil. The distinction is made by who is using it and how they use it that makes the final determination.

Chapter 5

Future Work

Future work for this thesis will include the addition of new attacks as researchers identify them and the implementation of attacks covered in this thesis to examine their performance as networks, such as Tor or I2P, grow. For example, can a congestion attack that requires probing every node in the network survive if Tor reaches 100,000 nodes? Perhaps a 10 minute life for the circuit is too short of a duration for this attack to survive. There is also potential for further classification of the attacks presented in this thesis. We briefly touched on the subject of denial of service attacks. While this is not a direct attack on a user's anonymity, it can have a significant effect on the anonymizer network and this can be a category on its own.

Bibliography

[1] "Metasploit." Internet: www.metasploit.com, April, 2012.

[2] Michael Hale Ligh et. al., *Malware Analyst's Cookbook*, Indianapolis, IN: Wiley, 2011, pp. 2-10.

[3] Kevin Poulsen (2007, July). "FBI's Secret Spyware Tracks Down Teen Who Made Bomb Threats." Internet: http://www.wired.com/politics/law/news/2007/07/fbi_spyware, July 18, 2007 [December, 2011].

[4] "Microsoft Security Bulletin MS07-017." Internet: http://technet.microsoft.com/en-us/security/bulletin/ms07-017, Dec. 09, 2008 [December, 2011].

[5] "Microsoft Security Bulletin MS06-001." Internet: http://technet.microsoft.com/en-us/security/bulletin/ms06-001, Jan. 05, 2006 [December, 2011].

[6] "Metasploit Decloaking Engine." Internet: http://decloak.net, [March, 2011]

[7] "Hacker Builds Tracking System to Nab Tor Pedophiles." Internet: http://www.zdnet.com/blog/security/hacker-builds-tracking-system-to-nab-tor-pedophiles/114, Mar. 6, 2007 [February, 2011].

[8] "Tor Project: Anonymity Online." Internet: https://www.torproject.org/, [April, 2011]

[9] R. Dingledine and N. Mathewson. Tor: The second-generation onion route. In Proceedings of the 13th USENIX Security Symposium, August 2004.

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

[11] X. Fu, Z. Ling, J. Luo, W. Yu, W. Jia, and W. Zhao. One cell is enough to break tor's anonymity. In Proceedings of Black Hat DC, February 2009.

[12] Arma. "One cell is enough to break Tor's Anonymity." Internet: https://blog.torproject.org/blog/one-cell-enough, Feb. 19th, 2009 [Feb. 2011].

[13] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell counter based attack against tor," in Proceedings of 16th ACM Conference on Computer and Communications Security (CCS), 2009.

[14] X. Wang, J. Luo, M. Yang and Z. Ling, A novel flow multiplication attack against Tor. Proceedings of the 13th IEEE International Conference on Computer Supported Cooperative Work on Design, 2009.

[15] O. Berthold, H. Federrath, and M. Kohntopp. Project anonymity and unobservability in the Internet. In CFP, pages 57–65, New York, NY, USA, 2000. ACM.

[16] A. Back, U. Moller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In I. S. Moskowitz, editor, Information Hiding (IH 2001), pages 245–257. Springer-Verlag, LNCS 2137, 2001.

[17] C. Li, Y. Xue, Y. Dong, and D. Wang. "Super nodes" in Tor: existence and security implication. In Proceedings of the 27th Annual Computer Security Applications Conference

[18] M. Wright, M. Adler, B. N. Levine, and C. Shields. Defending anonymous communication against passive logging attacks. In IEEE Symposium on Security and Privacy, pages 28–41. IEEE CS, May 2003.

[19] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource routing attacks against anonymous systems. Technical Report CU-CS-1025-07, University of Colorado at Boulder (2007)

[20] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naıve-bayes classifier," in CCSW '09: Proceedings of the 2009 ACM workshop on Cloud computing security. New York, NY: ACM, November 2009, pp. 31–42.

[21] Y. Shi, K. Matsuura, C. Mitchell, G. Wang, Fingerprinting attack on the Tor Anonymity System. In Information and Communications Security (2009), pages 425-438.

[22] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In Proceedings of the 2005 IEEE Symposium on Security and Privacy. IEEE CS Press, May 2005.

[23] N. S. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on tor using long paths. In Proceedings of the 18th USENIX Security Symposium, August 10-14 2009.

[24] S. Chakravarty, A. Stavrou, and A. Keromytis, "Traffic analysis against low-latency anonymity networks using available bandwidth estimation," in Proceedings of ESORICS, 2010.

[25] H. Sengar, Z. Ren, H. Wang, D. Wijesekera, and S. Jajodia. Tracking skype voip calls over the internet. In Proceedings of INFOCOM '10. IEEE, 2010.

[26] Wang, X., Chen, S., and Jajodia, S. Tracking anonymous peer-to-peer voip calls on the internet. In Proceedings of the ACM Conference on Computer and Communications Security (November 2005), pp. 81–91.

[27] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "DSSS-based flow marking technique for invisible traceback," in Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P), May 2007.

[28] P. Manils, A. Chaabane, S. L. Blond, M. A. Kaafar, C. Castelluccia, A. Legout, and W. Dabbous. Compromising Tor Anonymity Exploiting P2P Information Leakage. Technical report, INRIA, 2010.

[29] S. Le-Blond, P. Manils, C. Abdelberi, M. A. Kaafar, C. Castelluccia, A. Legout, and W. Dabbous, "One bad apple spoils the bunch: Exploiting P2P applications to trace and profile tor users," CoRR, 2011.