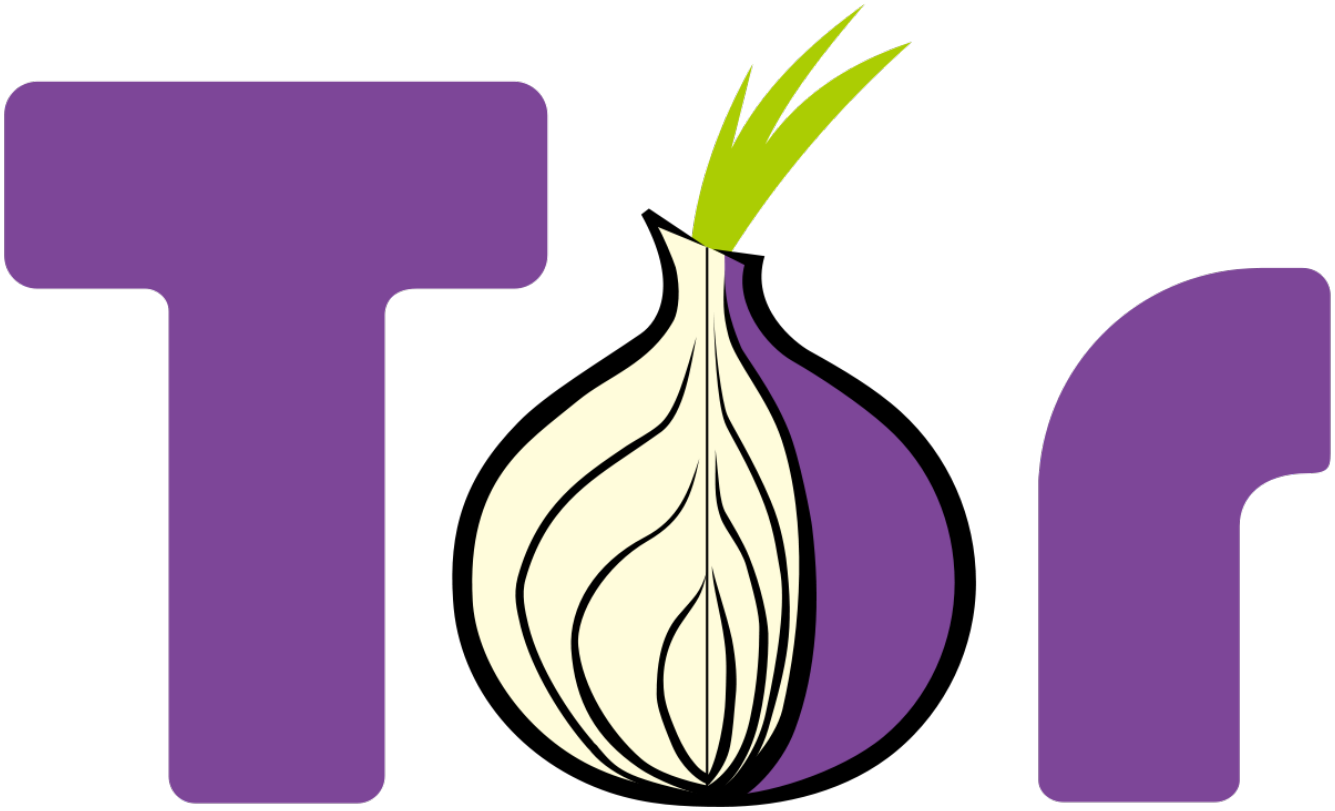
Tor Network
Technical Overview



Prepared by:

Simon Owens

Date: April 26, 2018

Course Number: CS-415

1. Summary

1.1. Objective and Scope

This paper will cover the technical aspects of what the Tor Network is, how it works, and why it is considered secure. Threats to the Tor Network will be briefly covered, but just minor speculation. The moral and ethical implications of those using the Tor Network should be covered in an entirely different paper. For more information on the design of the Tor Network, visit the following page.

1.2. Executive Summary

Tor is a great tool for avoiding censorship and traffic analysis. The tool isn't a fix-all solution to staying anonymous on the Internet. A virtualized private network (VPN) should be used when running Tor, and user's should follow Tor's best practices on their site. They have great documentation on implementation, high level design explanations, as well as access to the source code. Many members of the Tor community have developed threat models, penetration tests, and research to make the network as secure as possible. Tor developers have fixed several vulnerabilities in their browser and network to keep it secure. The Tor browser and network continues to develop at a fast rate and is the best solution for anonymous browsing and hosting.

2. Tor Overview

The Tor Network was founded by Roger Dingledine and Nick Mathewson. The Tor Project, the development of the open source tools utilized in the Tor Browser and Tor Network in conjunction. The Tor Network is an open network that attempts to keep users uncensored and anonymous in regard to the other people that they may interact with on the Internet. This means that anyone listening to your network traffic doesn't know what information is being sent or received, where it is going, and how much is being sent or received. A users traffic is bounced around to various places on the Internet before being directed to the requested site. The Tor Browser is an application that allows users to employ standard websites as well as onion websites, hiding all of the complexity of "bouncing around

traffic" to remain uncensored and anonymous. Onion Services are like standard websites, but only the Tor browser can access the information.

Tor users are able to bounce their traffic around the Internet by connecting to various virtual tunnels called Tor Nodes. These nodes are a virtual machine that acts as a router for the Tor Network. Below is an illustration of Internet traffic routing through the Tor Network. The Tor Browser obtains a list of Entry Nodes from the Tor Database.

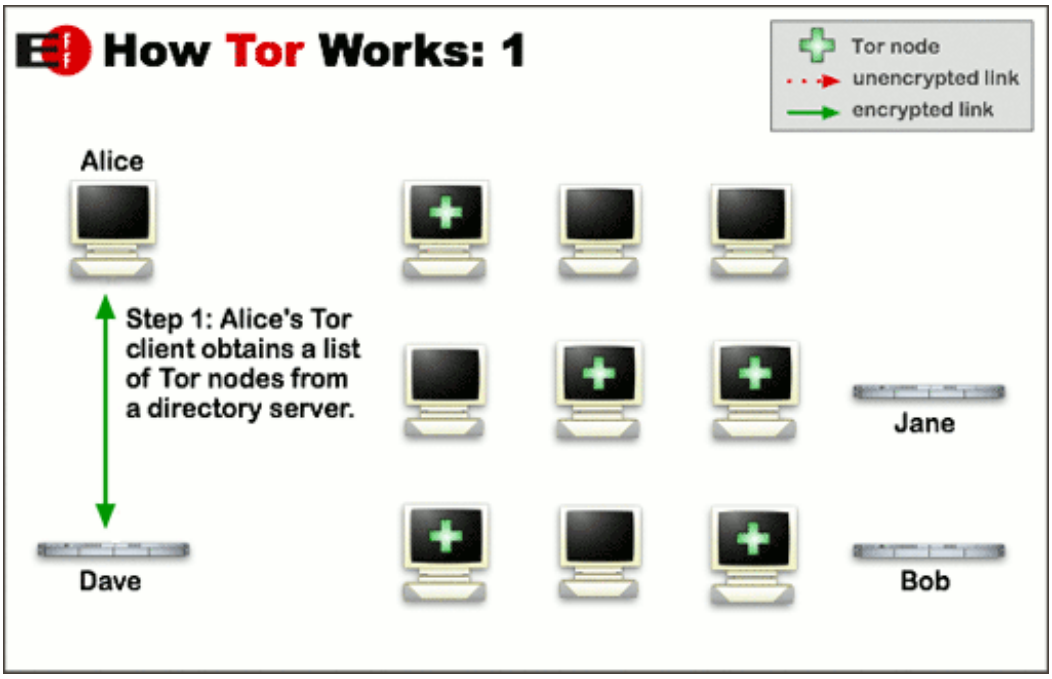


Figure 1: Tor Searches for Entry Nodes [1]

A connection with the first node is established with encryption. Then, that node looks for other Relay Nodes. A separate encrypted session is then established with the Relay Node. The original message is now encrypted twice, with two different keys. Each Relay Node only knows its next and previous node. The Middle Relay only knows which node sent it traffic, and which node it should forward the traffic. This is why the Tor Network is sometimes called the Onion Network, because it has multiple layers of encryption to peal back before getting to the actual message.

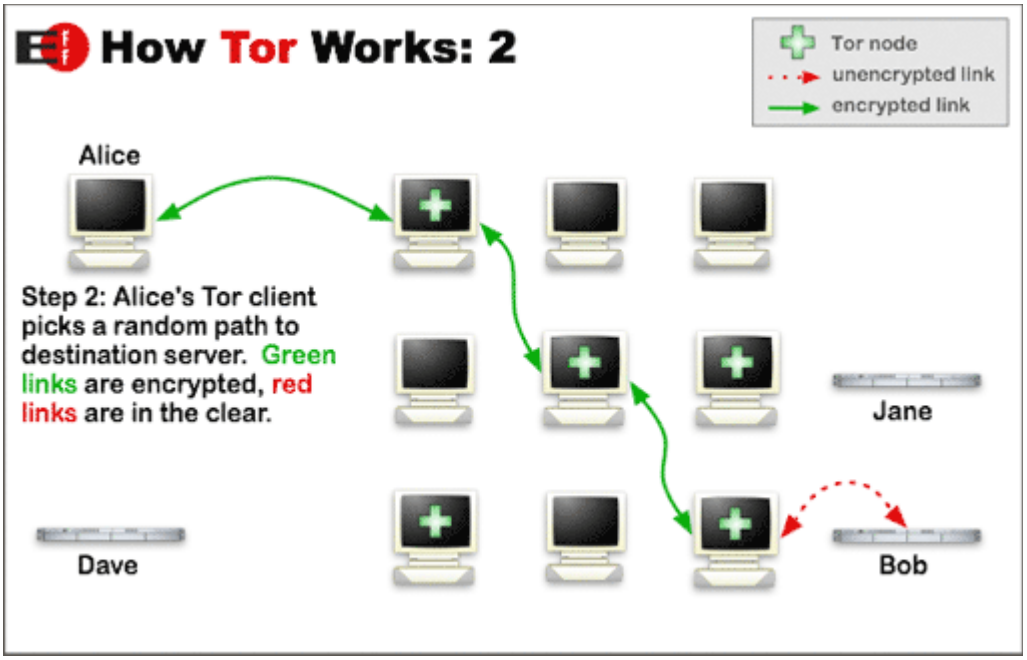


Figure 2: Tor Incrementally Builds Circuit [1]

If the user wants to browse another site, they will have to restart the process. They will look for another entry node, and incrementally build a circuit. Anyone analyzing all of this network traffic does not get any indication where Alice is browsing.

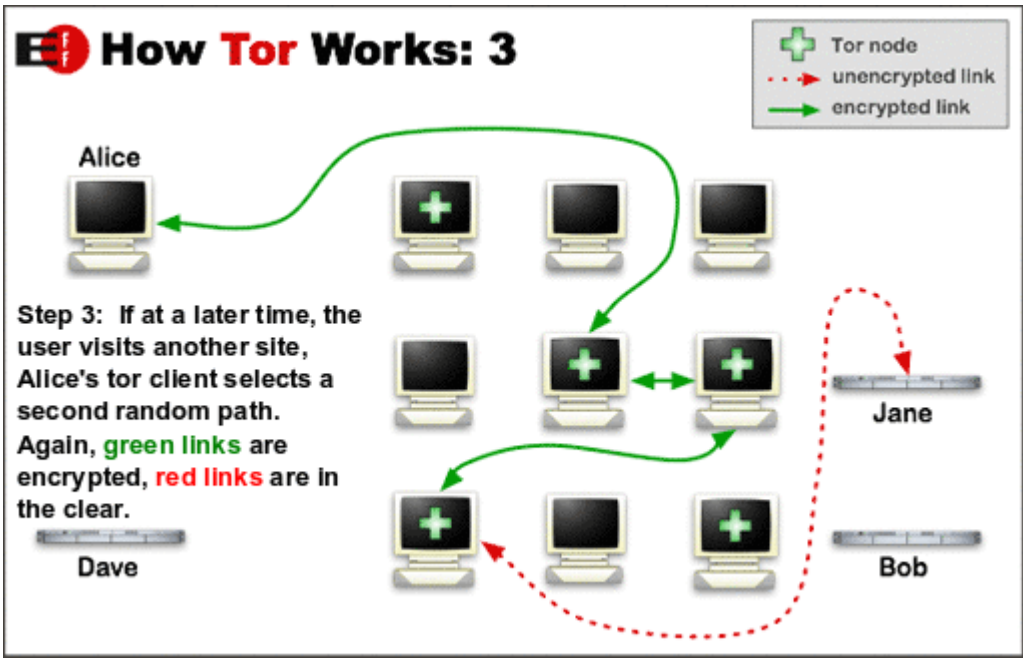


Figure 3: Circuit Rebuild Each Connection [1]

There are four different types of Tor Nodes: Guard Node, Middle Node, Exit relay, and a Bridge. **Guard(Entry) Node:** This is the first connection the Tor browser will make when attempting to browse the web. Those nodes are stored in the Tor Database so the browser knows what it can start it's connection with. **Middle Node:** This acts as the second connection in a Tor circuit. It neither knows the address of the original user, or what the user is attempting to connect to. This node only knows the entry and exit nodes. **Exit Node:** This is the final node in the Tor circuit. This node makes the request to whatever the user originally requested. The sites they are requesting will see this IP address, instead of the user's. **Bridge Node:** All Tor nodes are publicly available except for bridge nodes. The addresses of these nodes stay secret. Some Governments, ISPs, and websites block connections coming or going to Tor nodes, but Tor bridges stops this. For example, I downloaded the Guard Node from the Tor Website, uploaded it to my home server, started an Ubuntu docker container, configured system settings, and opened network ports on my firewall in under a minute. Traffic immediately starting routing in and out of my Tor Node! Anyone scanning my firewall can now see that I am hosting some type of Tor node. Analyzing the traffic with Wire Shark did not real any information on what type of traffic was going through the Tor Node.

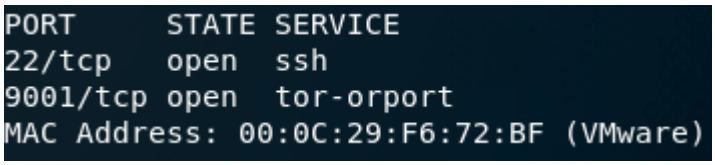


Figure 3: Nmap Scan of external Firewall

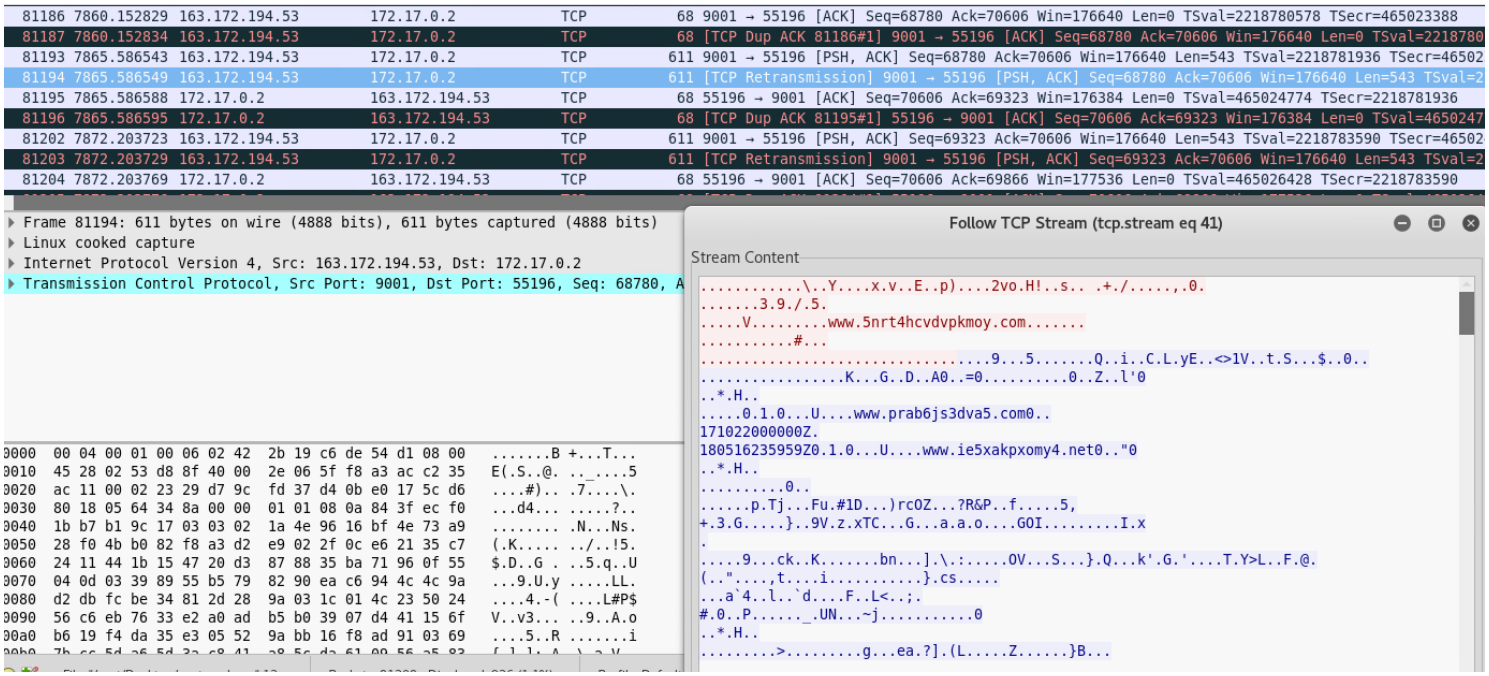


Figure 4: Tor WireShark Packet Capture

Only the Source of the information was visible to me, the rest of the traffic was encrypted. I found the general location of the person that was routing traffic through my entry node.

Figure 5: IP Geolocation

I then downloaded and ran the Tor Bridge. Again, I could only tell where the traffic was coming from, and where it was being sent. I did not run an exit relay. There are legal implications for running an exit relay. If a user were to download copyrighted information through my Tor Exit Node, the IP address would be associated with my home.

Hosting Tor nodes is not required at all to use the Tor browser, many people do not host nodes but use the Tor browser. The Tor browser is just a fork of Mozilla's Firefox, so it is extremely easy to

use. To visit normal sites, just search for them like a normal web browser. Sniffing the firewall's traffic on my local network I can see a normal HTTPS key exchange when attempting to browse Facebook. Figure 6 is an abstraction that I can see from a WireShark packet capture.

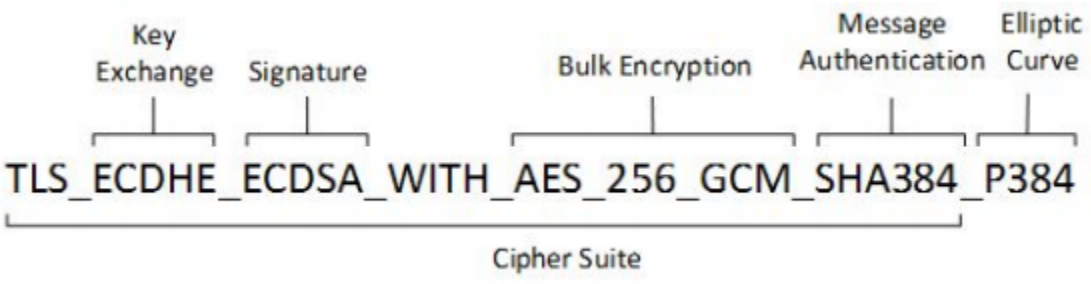


Figure 6: Cipher Suite String Structure [11]

This encryption currently has no weaknesses and would take a large amount of resources to break. However, accessing Onion services is different. Onion pages, which can only be accessed through a Tor Browser, do not have an associated IP address with them. Refer to section 4.2 for an explanation of how Tor's DNS works for onion services.

3. ISP and Surveillance Overview

To truly understand how Tor works well and why it is used, one must have an intimate understanding of the Internet and its workings. The Internet is just a large collection of computers that can communicate to each other. Computers can communicate with each other through various protocols to accomplish various tasks: file transfer protocol(FTP), secure shell(SSH), domain name service(DNS), and Hypertext Transfer Protocol (HTTP/HTTPS) which is used for browsing media on other computers. The collection of computers that accept HTTP/HTTPS connections is called the Web. The collection of computers that have various services open, but that you cannot navigate to, are called the Deep Web. The collection of computers that expose services, similar to HTTP, that need certain keys, passwords, or other authentication is called the Dark Web. It is important to be able to distinguish all of these different types of networks.

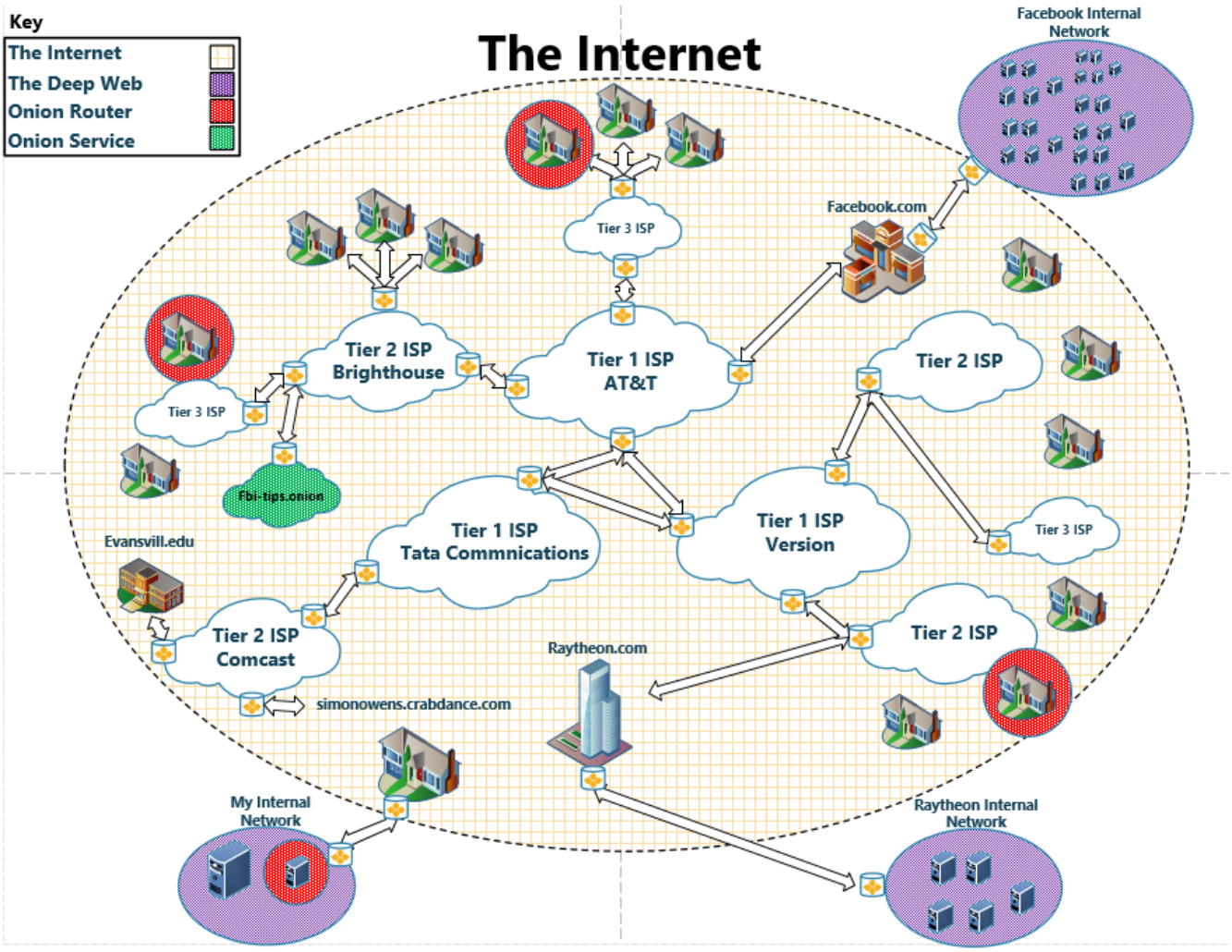


Figure 7: Internet vs Deep Web vs Dark Web

Internet Service Providers (ISPs) are corporations responsible for building and managing connections so that users can connect to any computer publicly available. Large ISPs, like AT&T, build cross continental pipelines that can transfer s amounts of shielded electrical or optical current. They collaborate with other ISPs to move Ethernet traffic around the world so that everyone can communicate with each other. There are various Tiers of ISPs based on how many connections the ISP has and can manage. All Tiers of ISPs can provide Ethernet connection to businesses and homes, but it is common for small ISPs, Tier 3, to provide Ethernet to several homes, towns, etc, and then forward all of that traffic onto larger ISPs. Figure 7 is a scenario where I pay a Tier 2 ISP. When I attempt to connect to Facebook, this ISP searches what it can connect to, realizes it cannot connect to Facebook

directly, so it gives this request to a Tier 1 ISP.

Since ISPs handle your Internet traffic, they analyze and record it so they can better optimize routes, estimate hardware upgrades/requirements, and various other things. They also enforce rules onto their customers so that they have the appropriate bandwidth and usage. In some instances, the ISP might block or slow down connections to sites for various reasons. This is currently happening in China and Iran. They have a way of blocking or throttling social media and news sites, as expressed in Figure 8. With Tor, users can get around these blacklists and firewall rules.

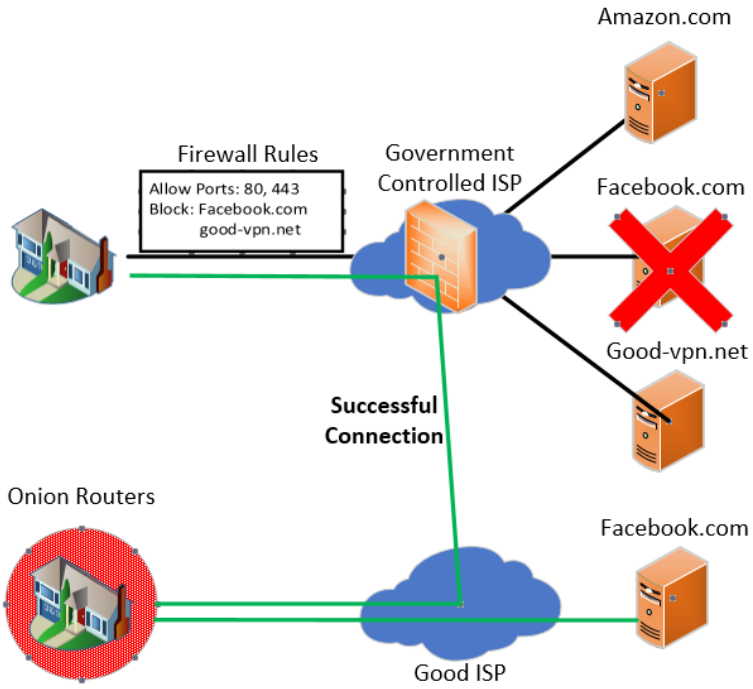


Figure 8: ISP IP blocking

Iran users can still get to Facebook with Tor because the Onion routers their Tor Browser connections to can navigate to Facebook. There is current controversy and changes happening in the United States with the Battle for the Internet. Some ISPs may also sell the information they collect to other companies for advertising, insurance, or various other reasons [9]. Security and governmental agencies have traditionally worked with ISPs in attempt to monitor illegal activity and to help gain additional information about criminal/cases through the sharing of metadata [10].

Metadata is data that provides information about other data. There are various reports that the National Security Agency is teaming with Verizon, as well as other ISPs, to collect metadata and build profiles for clients. This is just one example, and I won't search for anymore information on what Metadata they've collected in the past, because that information is classified. So what information can ISPs divulge when sniffing your network traffic and how much effort would it take them to analyze everything you are sending? I will use browsing Facebook on google chrome as an example. Connecting to Facebook with google chrome makes you use an HTTPS connection. This connection is secure with a 2048-bit RSA key which has no known weaknesses at the moment. Because of the encryption HTTPS uses, the only data I can see are: Time, source, destination, protocol, length, and a pieces of miscellaneous info. It would take a tremendous amount of effort to break this encryption. Even with the CIA and NSA's huge black budget to conduct surveillance, close to 60 billion dollars, breaking several key exchanges would be way too expensive [2]. Unless security agencies have a much better way than the Pohlig-Hellman algorithm to brute force keys, then they can only break a select few of high priority targets. This means that security agencies don't record the contents of all of one's web connections. Instead of speculating on what could potentially be possible, I attempted bruteforcing keys and conducting deep packet inspection on the traffic I was receiving through my Tor Nodes. My Unifi USG doing deep packet inspection on my network can tell me quite a bit about what my LAN traffic is and where it's going.

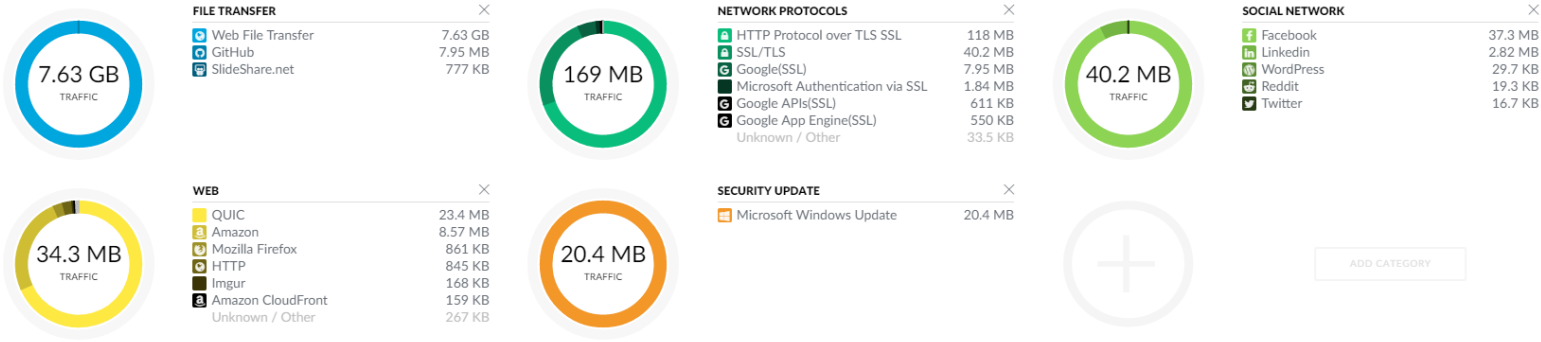


Figure 9: Deep Packet Inspection of Standard Traffic

It tells me what network protocols are being used, where it's going, and how much of it there is. Capturing Tor traffic on the other hand reveals different results.

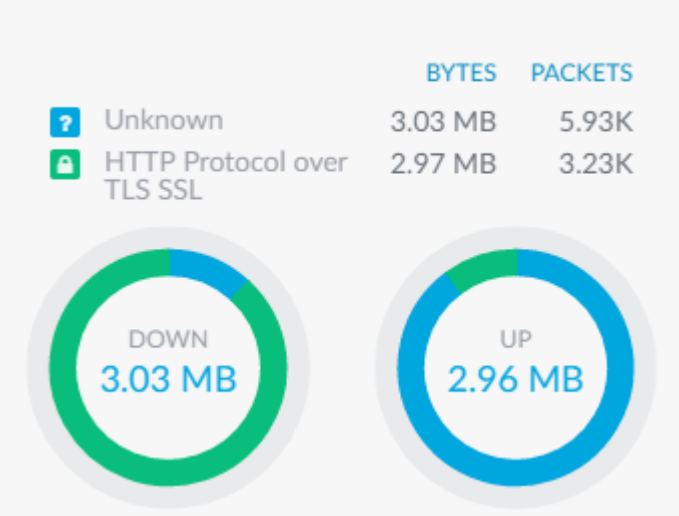


Figure 10: Deep Packet Inspection of Tor Traffic

No matter what, a constant stream of traffic always follows through the Tor node. The packets all have the same size, and same packet header.

TCP	68
TCP	68
TCP	611
TCP	611
TCP	68
TCP	68
TCP	611
TCP	611
TCP	68

Figure 11: Tor Packet Size

I could only see where it's coming from, if I am the entry node.

I then tried brute forcing one of the key exchanges I captured. I gave my virtual machine 32GB of ram, and 12 cores from two Xeon X5680 (3.33 GHz) to attempt to brute force one of the AES-128 keys. I let it run for a day straight with no success. Most all attackers will not waste electricity and run their server's at maximum capacity like I did to break standard encryption. All of this information

doesn't matter if your Tor traffic uses unencrypted protocols like FTP, Telnet, or HTTP. As the entry node, I could have easily seen all of this information they were sending to me. After capturing traffic for one day straight, the only unencrypted traffic I found was nmap traffic. I will not provide IP addresses of those that were conducting nmap scans. I did not find any other unencrypted streams that went through my entry node. If I was a relay node(in the middle), I would have not be able to analyze the traffic because it has another level of encryption from the Tor network.

Now that we've see how the Internet and Tor works, I will give some examples of what happens when people use the Tor browser. I typed in Facebook.com on the Tor Browser and was immediately taken there. My Tor browser had to connect to an entry node, relay, and exit node before being directed to Facebook's website. In the example below, the red circles are Tor nodes. The red circles marked with numbers are as follows:

1. Tor entry node
2. Relay
3. Exit node

Each of these nodes had their own established connection with a key exchange to encrypt traffic to each other.

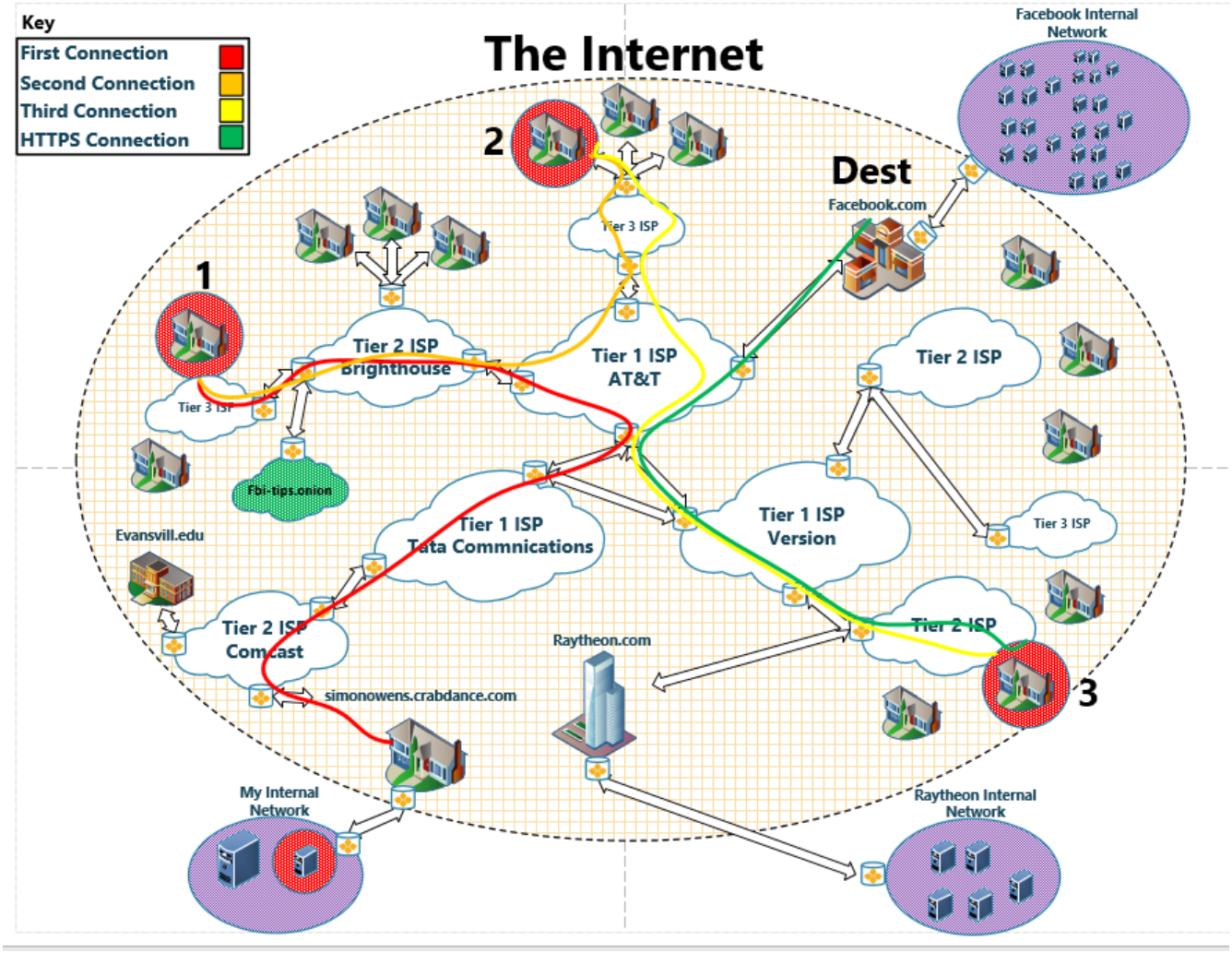


Figure 12: Tor Traffic Routing

First my browser is directed to 1, then 2, then 3, and then finally to facebook. Multiple ISPs were used, multiple times, in order to browse to facebook. My ISP, Comcast, doesn't know that I'm visiting facebook, they just know I'm sending HTTPS traffic to the entry node over port 9001 if they are doing deep packet inspection. Comcast can't make an informed guess on where I'm browsing to or what protocols I'm using, like file transfer, they can only see encrypted packets being sent to node 1 at a constant rate.

In conclusion, ISPs are monitoring your traffic and building profiles on users based on what sites they visit, when, and how much they browse. It may store browsing information if users utilize unencrypted protocols. Using Tor makes it much more difficult for ISPs to record what you are using

your bandwidth on.

4. Technical Analysis

4.1. Entry and Exit Nodes

Part of what keeps the Tor network secure is how the entry and exit nodes are chosen. Tor has strict specifications when building paths for users. The exit node is chosen first, and then the rest of the path is chosen before the circuit is actually built. Entry and exit nodes can be customized by selecting which ones you want to use in the public directory, but this isn't recommended as it lowers the amount of nodes you will use. When the Tor client requests a site, the Tor network attempts to build a circuit based on bandwidth, minimum hop distance, and a variety of other factors [7]. Because of the somewhat random nature of the nodes selected, along with the amount of nodes on the network, it is highly unlikely for an attacker to control both an entry and exit node to attack the client. Nodes that maliciously alter traffic are reported, and then blocked when users attempt to create a circuit. There are strict guard rules that must be met before a circuit can be built. The following criteria is used to find and stop attackers:

- The same router can't be used twice in a path
- Only one router is used in a /16 network
- The first node must be a guard node that has been validated with a public onion key
- Nodes that cause invalidation from public keys are reported to the tor developers

If an entry node is compromised, Tor circuits will avoid using them [8]. Clients have been comprised in the past from timing attacks by their ISP and surveillance agencies that can monitor traffic going to the entry and exit nodes. To add another layer of security to stop timing attacks, users should connect to a VPN before using the Tor browser. Attackers would have to break the VPN key or compromise the VPN provider to conduct a timing attack on the Tor user's traffic.

4.2. DNS and Key Management

How Tor's DNS works was abstracted earlier in section 2 because of its complexity. Tor's DNS plays a huge role in what keeps it secure. Tor users can connect to Tor services without the Tor service revealing it's IP address to anyone. This is done through public key cryptography. Each onion service collects the entry nodes that can reach it and generates a public/private key pair. The public key is combined with the entry nodes, and generates a type of certificate with this information that is called an onion descriptor. This onion descriptor is signed with the corresponding onion service's private key, so that it can later be validated by clients. Then each onion service establishes communication to the onion database, and gives it this onion descriptor that acts as a public key.

Each time a Tor client requests to visit an.onion site, it first contacts the onion database, which, in turn, searches it's database for that onion descriptor. The onion database supplies the client with the onion descriptor. This means that the Tor client must have already known the Tor service's.onion address ahead of time. One example of Tor address listings can be found on: https://thehiddenwiki.org/. Now that the Tor client has the public onion descriptor, it picks a random entry point, called a Rendezvous point, and establishes a secure session with a one-time pad. Once this connection has been established, the Tor client provides the Rendezvous point the onion descriptor, which then looks at the onion descriptor to find nodes that know how to reach the onion service. The Rendezvous point establishes a Tor circuit to one of those nodes. The Rendezvous point then successfully delivers the Tor client's request to the onion service. Any message the Rendezvous point receives from the onion service is validated with the onion descriptor. The Tor client and onion service now has a circuit to communicate back and forth with each other. This circuit consists of 6 relays total: 3 picked by the Tor client, and 3 picked by the onion service. No single node is responsible for a client or onion service. No single relay knows the client or service's address either. If the communication

stream was somehow compromised somewhere, the client would reject the message because validating the message with the public key would show that the message did not come from the onion service [5]

4.3. Key Exchanges (Asymmetric Encryption)

The Tor network uses RSA 1024-bit keys for their public key encryption. A fixed exponent of 65537 is used with SHA-1 as a digest function. Their key exchange uses the diffie-hellman protocol with a generator of g, and the primes come from rfc2409 section 6.2 which are considered cryptographically safe primes [4]. This document, rfc2409, details a framework that was created for generator cryptographically strong prime number for authentication and encryption. The group of primes that was chosen for the elliptical curve encryption is based on Galois Field (2^185), with the order of "0X01ffffffffffffffffffffffdbf2f889b73e484175f94ebc" which is represented in hex since it is so large [3]. This differs from the National Institute of Standards and Technology (NIST) approved elliptic curves. Not using the NIST elliptic curves could be a security weakness, but the designer of the Tor network seem to be concerned that the NIST elliptic curves could be broken by the authors that created it, eg NSA or NIST.

4.4. Symmetric Encryption

Referring to Tor's specification document, 128-bit AES encryption is used for symmetric encryption streams. Connections to tor nodes use TLS/SSLv3 which is strong encryption. Stronger encryption can be used if it is supported by both the client and destination [4]. Both 128-bit and 256-bit AES are considered secure and there are no current weaknesses in those encryption algorithms. It would take a tremendous amount of resources to break unless the attackers knew how to solve the discrete logarithm problem.

4.5. Traffic Obfuscation

Earlier when conducting deep packet inspection on Tor, it was easy to tell what Tor traffic was. The contents of the Tor traffic was unknown, but the Tor traffic itself had a distinct signature, it looked

nearly the exact same every time. Some countries doing deep packet inspection on their Internet traffic have blocked Tor traffic. To stop this, Tor developed Pluggable Transports. This software obfuscates Tor traffic to make it appear like normal TLS traffic. Pluggable transport first sends an encrypted message to a reverse SOCKS proxy to setup the traffic obfuscation. Once this handshake is completed, the Tor client stands the reverse proxy it's desired messages that are have been scrambled by a random obfuscation algorithm based on the clients state. The responding keys are also sent to unscramble the traffic and to know where to forward that traffic onto. There are various Pluggable Transport developers and versions which makes correlating Tor traffic signatures extremely difficult since the ISP cannot break the TLS encryption used. The documentation and code is available for further examination [6].

5. Conclusion

Tor is a great tool for avoiding censorship and traffic analysis. The tool uses approved and strong cryptographic protocols for key exchanges and symmetric encryption. Tor also uses cryptographically strong primes and large keys when using it's key exchange and encryption algorithms. The source code and documentation is available online which has also resulted in tons of testing by the community and other security organizations. There have been several threat models, exploitation attempts, and research done to make Tor as secure as possible. There will continue to be security weaknesses in Tor throughout it's existence but that is the case with every piece of software developed. The Tor developers have responded to outside research, and continue to make improvements and fix vulnerabilities to keep the network secure. Tor is extremely easy to use and has great performance for being a distributed network of routers. Tor developers also made it extremely easy to contribute to the Tor community by running nodes. The tool isn't a fix-all solution to staying anonymous on the Internet. A VPN should be used when using Tor, and user's should follow Tor best practices on their site. As of 2018, the Tor browser is the best solution for avoiding censorship and

keeping user's safe from traffic analysis.

Appendix A: Sources and References

1. Inc. "Tor." *Tor Project: Overview*, www.torproject.org/about/overview.html.en.
2. Nelson, Steven, and Evan Vucci. "Trump Sends Congress Record High 'Black Budget' Spy Funding Request." *Washington Examiner*, 27 Feb. 2018, www.washingtonexaminer.com/trump-sends-congress-record-high-black-budget-spy-funding-request.
3. "The Internet Key Exchange (IKE)." *IETF Tools*, tools.ietf.org/html/rfc2409#section-6.4.
4. *Tor-Spec.txt - Torspec - Tor's Protocol Specifications*, gitweb.torproject.org/torspec.git/tree/tor-spec.txt.
5. Inc. "Tor." *Tor: Onion Service Protocol*, www.torproject.org/docs/onion-services.html.en.
6. *Pt-Spec.txt - Torspec - Tor's Protocol Specifications*, gitweb.torproject.org/torspec.git/tree/pt-spec.txt.
7. *Path-Spec.txt - Torspec - Tor's Protocol Specifications*, gitweb.torproject.org/torspec.git/tree/path-spec.txt.
8. *Guard-Spec.txt - Torspec - Tor's Protocol Specifications*, gitweb.torproject.org/torspec.git/tree/guard-spec.txt.
9. Snider, Mike. "ISPs Can Now Collect and Sell Your Data: What to Know about Internet Privacy Rules." *USA Today*, Gannett Satellite Information Network, 5 Apr. 2017, www.usatoday.com/story/tech/news/2017/04/04/isps-can-now-collect-and-sell-your-data-what-know-internet-privacy/100015356/.
10. Angwin, Julia, et al. "AT&T Helped U.S. Spy on Internet on a Vast Scale." *The New York Times*, The New York Times, 15 Aug. 2015, www.nytimes.com/2015/08/16/us/politics/att-helped-nsa-spy-on-an-array-of-internet-traffic.html.
11. msdn.microsoft.com. (2018). Cipher Suites in TLS/SSL (Schannel SSP) (Windows). [online] Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/aa374757(v=vs.85).aspx