Recent Attacks On Tor

Juha Salo
Aalto University
juha.salo@aalto.fi

Abstract

Tor is an anonymous communication network [3]. If more users are becoming interested in their privacy, the need for such anonymous services might increase. The second-generation Onion Router design Tor and its previous designs seems to have been under research and there have been rather recent papers on Tor's vulnerabilities. We will discuss the current attacks on Tor and make an effort to categorize them for further analysis.

KEYWORDS: vulnerabilities, security, onion routing, Tor

1 Introduction

According to [4] anonymity system Tor has an estimated over 250,000 users and thousands of network relays. Tor [3] is used for various activities online, such as web browsing, file transfers and instant messaging. By the same authors, the anonymity is ensured by preventing attackers to link communication partners or multiple communications to or from a single user. Further, more practical goals of Tor include deployability, usability, flexibility and a simple design.

Terminology around Tor and Onion Routing is ambiguous. We follow Onion Router Website's [14] proposal. An initial design for Onion Routing [7] was proposed in year 1996, and this design era is called the generation 0. The generation 1 [16] ranges from the initial design to the proposal of Tor. Tor [3], also known as the second-generation Onion Router, originated during years 2002 and 2005.

Dingledine et al. [3] presents and explains Tor. Tor, the second-generation Onion Router is a protocol that intends to anonymize network traffic in a low-latency manner. The protocol provides many improvements over the old Onion Routing design, hence Tor is called as the second-generation Onion Routing. These improvements include perfect forward secrecy, sharing one circuit to many TCP streams, no traffic shaping, separation of "protocol cleaning" from anonymity, leaky-pipe circuit topology, congestion control, directory servers, variable exit policies, end-to-end integrity checking, rendezvous points and hidden services.

2 Tor

All of the information in this chapter including 2.1 Protocol and 2.2 Threat Model are based on Dingledine et al. [3], unless stated otherwise.

2.1 Protocol

The basic idea of this overlay network is to construct a circuit, which consists of onion routers (OR) that know only its predecessor and successor. User uses circuit to pass data through the Tor network anonymously, as seen in Figure 1. Data is wrapped in layers using symmetric cryptography and in each onion router as the data goes through, a layer is unwrapped by using a symmetric key and relayed forward. At the end of the circuit, onion router relays data to the intended destination. The destination is not required to run Tor related software.

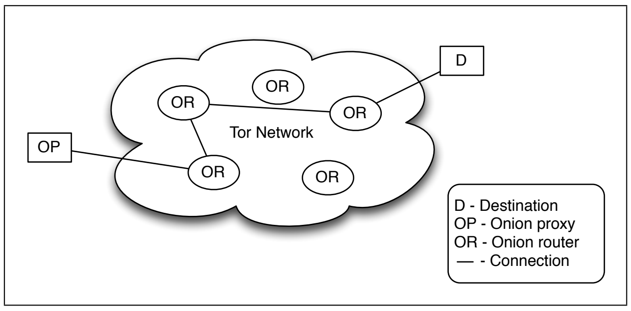

Figure 1: Overview of Tor [19].

Each onion router in the Tor network is connected to every other onion router using TLS. TLS is used to prevent possible attackers from being able to modify data, impersonate an onion router, and read the plaintext data by keeping the data secret in the connections. Tor users use onion proxy (OP) to receive directory information, create circuits in the network and manage user application connections.

Streams are multiplexed in a circuit, thus a one circuit can contain multiple TCP streams. Circuits are created preemptively by the onion proxies and rotated periodically to avoid traffic analysis. List of available onion routers that can be chosen to the circuit are downloaded from a signed directory service. Upon creating a circuit, the user's onion proxy negotiates a symmetric key with every onion router in the circuit, one by one as illustrated in Figure 2. Onion proxy always commands the last onion router in the circuit to extend one hop further until all intended onion routers are included. During this creation, onion router does not care who opens the circuit, but onion proxy knows the onion router. In other words, this handshake protocol is an unilateral entity authentication and provides forward secrecy. After the circuit has been established, relay cells can be sent.

A cell is a fixed-size, 512 bytes, unit of communication containing a header and a payload. Overview of cell struc-

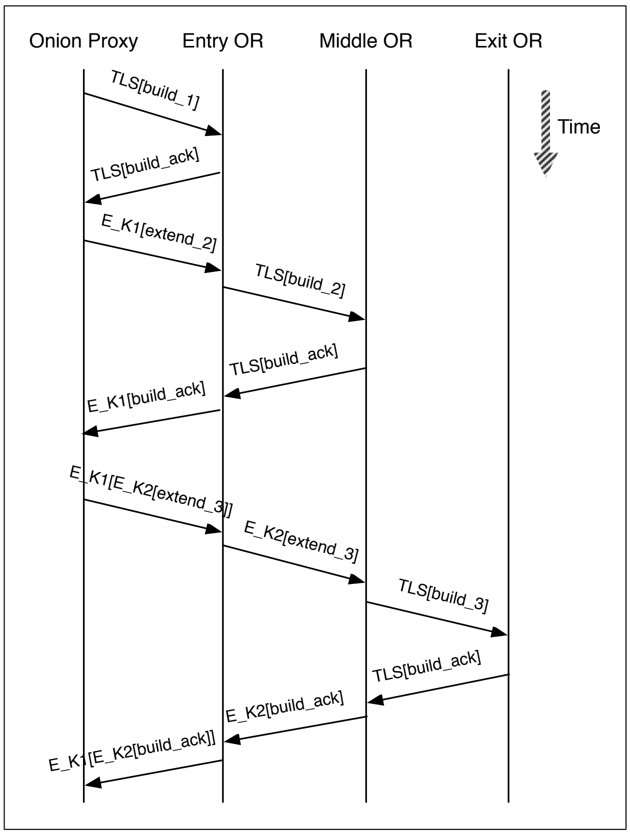

Figure 2: Circuit creation [2].

tures is presented in Figure 3. A cell is either a control cell or a relay cell. A control cell is handled by the recipient, whereas a relay cell contains end-to-end stream data.

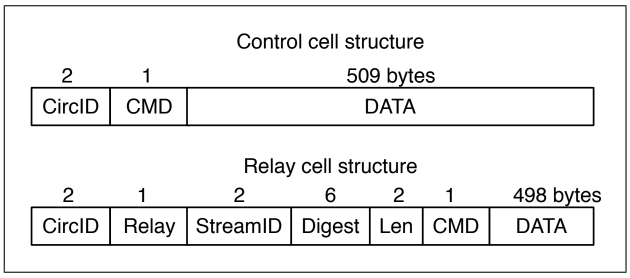

Figure 3: Cell structures [3].

For instance, when a user needs to send a packet through Tor, a relay cell must be created. Onion proxy creates relay cell in a iterative manner. Onion proxy calculates the digest, then encrypts the cell payload for every hop up to target onion router. In this method, only the target onion router will receive a valid digest after decryption. This topology is also known as leaky pipe, which means the stream's exit hop can be selected from the circuit, in contrast to having the last hop always be the exit point.

When an onion router needs to respond to the user, it encrypts the cell using a key negotiated during the circuit creation and sends the encrypted cell to the next onion router in the circuit towards the user. After the user's onion proxy has received the cell, it iteratively unwraps it. The received cell is decrypted using the keys shared with the onion routers from the nearest onion router to the last. Similarly with digest validation in the onion routers, the digest is validated in the onion proxy after each layer is unwrapped to ensure from whom the cell is from.

As a summary, in this protocol only the first onion router in the circuit knows the user's identity and the selected exit onion router is the only one who is revealed the destination. The onion routers in between only exchange encrypted information.

2.2 Threat Model

Tor does not protect against a global passive adversary. A global adversary means that the attacker can observe all links in the network. Tor's threat model assumes that the adversary has some fraction of the network under observation. Further, the adversary is able to compromise some onion routers, operate own onion routers, and delay, generate, modify, delete traffic.

Traffic confirmation attacks are not in the focus of Tor's threat model. Traffic confirmation attack means, that an adversary could confirm that two parties are communicating with each other over Tor by observing patterns, such as timing and volume of the traffic. Instead, Tor's increased focus is to prevent traffic analysis attacks, where attacker tries to determine in which points in the network a traffic pattern based attack should be executed. In other words, in this type of low-latency anonymous system the aim is to prevent attackers to know where to attack.

3 Attacks

3.1 Probabilistic models

The Bayesian Traffic Analysis of Mix Networks [18] introduces a probabilistic model for anonymous Mix networks that can be used to analyze the network, for instance measure the security. The model is flexible and different variables are taken into account. The model is currently a work in progress, and specially more research is required for popular low latency systems, such as Tor where in practice an adversary can only observer a fraction of the whole network.

Probabilistic Analysis of Onion Routing in a Black-box Model [6] follows on creating a model to evaluate how much an adversary can know about users based on a priori probabilistic behavior of the users. In this model, the design details are abstract, thus this model could be adapted to other anonymity networks other than Tor. Two assumptions are made, first, one user is responsible for one input and one output. Second, an adversary is able to link input and output to a user, if both of them are observable. In addition to the mathematical theorems, the results indicate that the worst user anonymity occurs when user becomes unique and identifiable when the user chooses a destination the others are unlikely to choose. Another occasion is when a user's choice is mistaken to a group's choice, when user chooses a destination the others prefer. In case of a common behavior and distribution, the anonymity tends to be the best possible.

Future research might focus on more detailed design decisions, such as entry guards impact on anonymity.

3.2 Entry and exit onion router selection attacks

Compromising Anonymity Using Packet Spinning [15] provides an attack that uses looping circuits and malicious onion routers. The looping phase in this attack aims to block other onion routers from being selected in circuits. Two assumptions are made: circular circuits are not detectable and a legitimate onion router will spend time in executing the cryptographic calculations. In other words, the malicious onion proxy creates loops in circuits to target onion routers to create a denial-of-service attack. If the looping phase attack is successful, then the malicious onion routers are more likely to be selected in circuits, because the other legitimate onion routers are busy. This advantage of the adversary can be used to execute further attacks.

Low-Resource Routing Attacks Against Tor [2] introduces an attack that benefit from lying resources information to directories. False information benefits the low-resource requirements set for this paper. Even though new versions of Tor have active measurements to validate resource information, such as router's bandwidth, it is interesting to evaluate what effects this attack has on Tor. In addition, it is possible to compromise onion routers with high bandwidth and uptime, thus avoiding giving false information by actually taking control of a target that has these resources. The false information can give the attacker an unfair fraction of requests for new circuits in the Tor network. These onion routers have increased probability to be chosen as entry and exit onion routers.

By [2], after ensuring that malicious onion routers are probable to be selected on both ends of the circuit, the next step in the attack exploits the victim's circuit creation process. The introduced attack exposes the circuit even before any payload data is sent from the user. The attack identifies the path by recognizing patterns in the Tor's circuit building algorithm. This attack in a simulated environment with 66 onion routers having 6 malicious onion routers compromised over 46 % of paths.

Bauer et al. [2] introduce methods for low-resource malicious onion routers to handle their faked position, for instance, to avoid bandwidth problems the malicious onion router only handles new circuits and if the malicious onion router detects to be in the middle of the circuit, then it can break the circuit to fasten the process of becoming selected as the first or as the last onion router. In addition, as the Tor network becomes more congested, the probability of a new circuit choosing a malicious onion router increases. A related attack is focused on entry guards: flooding the network with false router advertisement data could increase the threshold for choosing entry guards, thus limiting the number of correct entry guards becoming selected. In theory, it would be possible to displace all valid entry guards with malicious ones. This attack assumes that the selection of entry guards is based on some median on certain variables, including bandwidth and uptime. In other words, onion routers having better variables than average could become entry guards.

According to Bauer et al. [2], verification of advertised onion router data can help detecting faking onion routers.

3.3 AS and global level attacks

AS-awareness in Tor Path Selection [4] focuses on autonomous systems (AS) in relation to Tor. In more detail, Edman and Syverson [4] analyze the current potential threat of AS-level adversaries against the Tor network and provide evaluation of Tor's path selection algorithm with some suggestions.

As stated in [4], autonomous system is an independent network, and Internet consists of these ASes. For instance, when sending message using Tor, the traffic goes through multiple different autonomous systems. More importantly, if both the entry and exit onion routers are located at the same AS, then a statistical correlation attack can be performed on the AS-level.

Edman and Syverson [4] researched the assertion found in anonymity literature, which states that it is less likely for an AS to listen both the entry and exit onion routers when the Tor network grows, because there are more users from different places. The results show, that this is not completely accurate, since after a point the amount of ASes is balanced and new users occupy the old autonomous systems. The analysis indicated a drop from 2004's 38 % to 2008's 22 % in overall mean probability of a single AS having the opportunity to listen both ends of a communication. However, the change is a rather small since in the year 2004 the number of relays were 33, and during the year 2008 the amount of relays was about 1240. Also, further analysis indicated a significant negative impact on path diversity.

By [4], Tor's improved path selection algorithm has had improvement in preventing AS-level traffic analysis. These improvements include requiring onion routers to be from different /16 subnets and an entry guard limitation to entry onion routers. Future improvements in Tor's path selection algorithm could implement country based diversity requirements or a more theoretically effective way of approximating AS paths.

Large Scale Simulation of Tor [13] evaluates several global passive adversary attacks in a simulated environment. The environment was build up using SSFNet event based simulator and the test-network consisted of 6000 onion routers. In the environment, three different global passive adversary attacks were performed. These include connection start tracking attack, packet counting attack and stream correlation attack.

As stated in [13], connection start tracking attack compares a stream entering a network and an event when a stream emerges from a network. However, because Tor multiplexes streams it is difficult to recognize when a stream starts and ends. Also, delays need to be taken in concern when evaluating the possible timeframe of stream entering and leaving the network. In the simulation, 98 % to 96 % of streams were eliminated from the pool of possible candidates for communicating partners.

By [13], packet counting attack counts the packets entering and leaving an onion router. With this information, streams can be detected by comparing the number of packets going through the onion routers. The non-eliminated streams from connection start tracking attack could be further eliminated using packet counting attack. The results of packet counting attack is explained very briefly. However, simple packet counting attack is considered being effective [3].

As explained in [13], stream correlation attack can be divided to fixed time window and peak extraction type of an attack. In a fixed time window attack fixed time is set. During this time, packets are counted and this operation is repeated until attacker has sequence of packet counts. This sequence is used to identify the stream in other observation point. In peak extraction attack a stream is divided into fractions and packets in these fractions are then counted. Peaks in the stream can be observer and this information can be used to identify the Tor stream. Fixed time window attack was highly effective, 80 % of the streams were identified in a low traffic network. The peak extraction attack is not as effective as the fixed time window, though peak extraction has advantages in networks with high delay values.

3.4 Traffic and time analysis based attacks

Low-Cost Traffic Analysis of Tor [12] presents an attack that includes traffic-analysis techniques and how initiator's otherwise unrelated streams can be linked back. The term low-cost comes means that the attacker is not required to be a global adversary, instead only a partial view of the network is assumed. Unlike in the the Tor's designers' beliefs, timing attacks are possible even with in limited threads model [19]. By [12], since even one extra connection on a Tor node results in higher load, an an attacker can connect through targeted tor nodes and measure latencies of the messages. Estimate of the traffic load of a Tor node can be analyzed against known traffic pattern using techniques of traffic-analysis.

According to Murdoch and Danezis [12], a variant of this attack contains a malicious server, that sends data to the victim in a pattern. This pattern is then observed by creating a connection through over the candidate onion routers and executing traffic analysis.

A cell counter based attack against Tor [10] introduces a traffic analysis based active watermarking technique that reveals the communication partners in a Tor circuit. Results gathered from a real Tor network indicate this attack is effective (with low latencies, detection rate is near 100 %) and difficult to detect. The test used a specific onion proxy with circuit including a malicious entry onion router and an exit onion router. In a normal case, this attack requires control of both the ends of the circuit by an adversary. However, an variation of this attack can be executed even if the attacker has only control of an exit onion router. In this scenario, attacker could create a man-in-the-middle attack between the entry onion router and the user.

According to Ling et al. [10], the basic idea of the attack is to embed a secret signal in the cell counter in the traffic, that is then recognized by another malicious node in the network to confirm the communicating parties. The secret signal could be a sequence of bits. The signal injection can be executed either on the exit onion router or on the entry onion router. The injection works by manipulating the number of queued relay cells on each of the onion routers. For instance, attacker might indicate three relay cells for '1' and one really cell for '0'. Due to network congestion and latencies, some variation could occur during transfer of these cells from onion router to another. However, the authors provide mechanisms to mitigate these issues.

Browser-Based Attacks on Tor [1] presents a time based attack that exploits browser behavior when tampering HTTP traffic. In this attack, two malicious onion routers are required: an entry onion router and an exit onion router. The evil entry onion router analyzes the user's traffic for time based patterns. The malicious exit onion router modifies the HTTP-traffic to contain HTML or javascript based code that generates calls to a malicious Web-server in recognizable time patterns. In this attack, it is not required to have both the malicious evil and entry onion routers in the same circuit. If the user leaves the browser open after it has received code added by a malicious exit onion router, then it is enough that at some point of time a new circuit is created by the onion proxy that contains attackers entry onion router.

By [1], traffic analysis is more difficult when there exists other traffic, thus the noise ratio is increased. According to the authors, this attack is first to take advantage of Tor's exit policies to create a clean circuit without disrupting noise. An exit policy defines to what IP-addresses and ports is the corresponding exit onion router willing to relay traffic. Some ports, such as SMTP 25 and file sharing ports 4661-5666, are accepted in a minor group of onion routers, it is effective to use one of these ports as the malicious Web-server port in the attack code executed in a browser. When there are less onion routers that serve certain ports, then the probability of a malicious onion router becoming the entry onion router is greater. This method assumes that onion proxy tries to open a new circuit if the current circuit does not support these less supported ports.

As stated in [3], Tor protocol accepts end-to-end timing attacks as one of the threats. By [17], currently Tor seems to implement entry guards. According to Abbott et al. [1], entry guards are routers that are selected randomly or choosing from a set of trusted nodes, and only these selected routers can act as entry onion routers. This changes the probability distribution, but still adding a new malicous onion router to Tor network increases the probability for malicous onion routers to become selected, if entry guards are randomly chosen.

By [1], defenses include disabling active content systems and the use of HTTPS-protocol. Measures in active content systems, such as disabling Javascript prevents Javascript based code execution. Also, in a situation where user turns Tor off and open Web-page could expose identity, an add-on software could disable this kind of vulnerable open Web-pages. However, disabling this kind of systems might have negative impact on user experience. HTTPS-protocol prevents man-in-the-middle attacks, in other words the exit onion router can not modify or view the HTTP tunneled over SSL if user uses this protocol to view Web-pages. In practice, HTTPS-protocol is not that secure, because users often accept self-signed certificates despite a Web-browser's warnings.

A Practical Congestion Attack on Tor Using Long

Paths [5] is an attack that reveals an entire path of a user in a modern Tor network. This attack combines exit onion router's HTTP-stream modification and selective onion router congestion attack. The path is then exposed by evaluating changes in latencies. Three assumptions are set in order to create a successful attack. First, onion routers do not add artificial delays when routing packets. Second, directory servers provide list of all onion routers. Last, users can establish circuits of arbitrary lengths.

By [5], the attack can be separated in three different phases, which include modification of HTTP-stream on the exit onion router, observing latencies and a congestion attack. An attacker must have a malicious onion router as the exit onion router on a victim's circuit. Then, an attacker modifies either a Javascript or HTML-based code, that sends requests to attacker's server. The requests are sent with predefined time intervals and contain local system time to ease monitoring latencies. It is worth mentioning, that Javascript-based modification is more transparent to user and popular third party software such as Tor Button plugin and Privoxy does not disable Javascript by default.

As stated in [5], a malicious Web-server stores and analyzes the requests received by the attack code. The Tor network is not idle, thus the latencies vary, so an attacker must calculate a baseline latency, which is then compared to the measurements received during a congestion attack.

In [5], the congestion attack is done by exploiting a flaw in Tor's path building design that allows arbitrary length circuits that loop certain onion routers. It is possible to construct a long circuit, that contains the same router repeatedly, since each onion router only knows its predecessor and successor. To exploit this principle, an attacker can include a target onion router in the circuit, then connect to two or more other Tor routers, after the target onion router can be included again. In target router's packet scheduling, the single circuit is seen as multiple different circuits. This congestion attack requires limited bandwidth from the attacker, and is thought as an effective denial-of-service attack as well.

According to Evans et al. [5], all onion routers are suspected for being in the victim's circuit, thus searching the routers using the congestion attack is challenging. In addition, Tor circuit switching occurs every 10 minute by default, so the time frame when attacker's controls the exit onion router is limited, which requires optimization of the attack. Optimization could be achieved by pre-building moderate size circuits, choosing reliable routers and estimating guard onion router candidates to reduce the number of suspected onion routers.

Evans et al. [5] notes that this attack worked in a real experiment conducted in real Tor network, further stating that it is practical and effective. Congestion attacks that rely on a single congestion circuit are not practical in a modern Tor network. Since the network fluctuations are more lively, the probability to false positives and false negatives is greatly increased. For instance, attack suggested by Murdoch and Danezis [12] did not work, because it was not possible to separate normal congestion from congestion caused by an attacker.

By [5], disabling active content systems prevents the use of a signal generator injected by modifying HTTP-streams. However, the long path congestion attack is a challenging problem. Limiting the path length would be the key to prevent this type attack. Current Tor design could implement a method method of tracking the length of a circuit. Even if circuit lengths would be controlled, this does not prevent attacker from constructing multiple circuits. For instance, attacker could create a circuit to another onion proxy, that creates a new circuit and so on. A full solution to this problem is not yet known.

How Much Anonymity does Network Latency Leak? [9] propose two attacks on low latency anonymous systems, such as Tor: passive linking attack and client location attack. Both attacks benefit from a malicious server observing the latencies of a connection over an anonymous network.

As stated in [9], passive linking attack requires user to create two connections to a malicious server or connect to a two different malicious server. After this first step, a common distribution of round trip times (RTTs) can be assumed that is used to recognize the user.

Also by [9], client location attack tries to measure the RTT between victim and the entry onion router. This attack uses a malicious Web-server and a malicious onion router. The malicious onion router enables a clogging attack to determine the onion routers in the victim circuit and their RTT by creating a circuit through them. After the RTT between the entry onion and victim is known, the attacker analyses RTT between candidate routers and the entry router, in order to gain information about the victim's location. Different methods could be applied to measure RTT between two hosts without cooperation of either, for instance using network coordinate systems and pinging or King technique [8].

In [9], both of the attacks were tested and the results indicate that these attacks work. However, there is a lot of room for improvements and optimizations. Much better results were gained, when user used an application that periodically sends timestamps to a server. In Tor's protocol, authors suggest that adding artificial delays might be unavoidable, if no new path selection algorithms are adapted.

Passive-Logging Attacks Against Anonymous Communications Systems [21] examines a predecessor attack and an intersection attack. The predecessor attack [20] provides probability values to reveal user's identity. By [21], in predecessor attack, the attacker logs communication suspected to be coming from the user and because in Tor circuits are closed and re-established over time, thus the user's address in the network is seen more often than others. This way, attacker can identify the stream originator over time.

According to Wright et al. [21], in intersection attack the adversary keeps a list of addresses that have been active when the victim has contacted his destination. Over time, the list size decreases and it is more probable to identify the victim.

3.5 Protocol vulnerabilities

Effective Attacks in the Tor Authentication Protocol [22] introduces a vulnerability in the Tor authentication protocol during concurrent runs. The Tor authentication protocol is used during the circuit initialization, when the onion proxy negotiates a shared key with each of the onion routers. Zhang

[22] states that Tor authentication protocol will lead to inconsistency in session keys among two communicating parties and it is not secure in concurrent environment. The attack works by interleaving messages of several Tor authentication protocol instances. Further, the author proves this by using a mathematical model. However, no practical results were provided. In addition, an improved version of the Tor authentication protocol is proposed, that solves this vulnerability.

On the risks of serving whenever you surf [11] focuses on an attack that reveals bridge's IP address by visiting certain Web-sites. Since the list of onion routers can be easily downloaded from the directory service, it is easy for an authority to block access to Tor by blocking the access to these publicly listed routers. To counteract this censorship, a bridge was introduced in December 2007. A bridge is a unlisted first-hop relay, served by Tor end-users. Because there is a great number of end-users, the censorship authorities face difficulties, since now they have to block all the publicly listed onion routers and these bridges. Bridges information sharing is limited to frustrate the gathering of all bridges in the Tor network. However, the information should not be too scarce, since a bridge's user should receive the bridge information somewhere in order to connect to it.

By [11], three bridge related architectural vulnerabilities make this attack possible: because bridges are easy to find, a bridge accepts connection when its operator (a person who operates the bridge) is using Tor, and bridge user's (a person who wants to use bridge) traffic interferes with the traffic originating from the bridge operator. The attack exploits these weaknesses, thus the attack scenario can be separated in three phases: bridge discovery, bridge winnowing and bridge confirmation.

As stated in [11], bridge discovery should be balanced. Bridge users should be able to obtain bridge information, but not too easily to prevent collection of bridge information for censorship purposes. Bridge descriptors are given by bridge authority, and limited by IP address range. In other words, certain ranges are only given certain distinct bridges during some period of time. The authors were able to gather many descriptors by using Tor's exit onion routers to get different sets each time. Also, a candidate bridge can be validated easily, since bridges do not try to hide that they are bridges. A simple test on a bridge's default ports is sufficient, thus a censorship party can poll even a quiet large range of addresses for bridges. These large ranges could be country or organization specific.

According to McLachlan and Hopper [11], bridge winnowing means that an adversary checks if a bridge operator is serving and surfing. In other words, since bridge automatically accepts connections when the operator is using Tor, we can assume that if the bridge is offline, then the operator is not using Tor. To confirm if the bridge is serving, an attacker could just connect to bridge using IP address and correct port number. The next step in winnowing is to correlate online/offline-status of bridges to a known information about victim. Such information could be for instance some time information of "touches" on a Wiki-page. By comparing the times of victim using some service, we compare it to bridges that were online during that time period. In this way, we can limit the possible candidate bridges to contain only the ones that were online. The test results of this phase suggest to avoid in a long term operating a bridge while using some pseudonymous forum.

By [11], bridge confirmation uses circuit clogging attack [12] to identify the bridge from the pool of candidate bridges. As stated in [11], a bridge operator can be confirmed without middleman router probing to be an originator of a connection by a circuit clogging attack that is slightly modified. Several mitigation methods were provided, however all of them require limitation in the service level provided to the bridge users.

4 Conclusion

We introduced recent work on attacks against Tor and made an effort to sort the attacks into five different categories:

- Probabilistic models [18, 6] aim to provide information about the network, for instance measurements of security and anonymity, based on mathematical models.
- Entry and exit onion router selection attacks [15, 2] increase the probability of an adversary's onion routers to be selected as entry and exit routers in the victim's circuit.
- AS and global level attacks [4, 13] require an adversary, which has access to a great portion of the network. It is worth mentioning, that Tor's threat model does not protect global passive adversary attacks [3].
- Traffic and time analysis based attacks [12, 10, 1, 5, 9, 21] observe and possibly interact with the Tor network for instance by creating distinguishable patterns to weaken anonymity.
- Protocol vulnerabilities contain two attacks [22, 11] that introduce weaknesses in the actual protocol design. First, there is a vulnerability in the Tor's authentication protocol, however the implications of this attack is unknown to us. The second attack exploits Tor's bridge-service, thus revealing the IP-address of a bridge.

References

[1] T. G. Abbott, K. J. Lai, M. R. Lieberman, and E. C. Price. Browser-based attacks on tor. In *PET'07: Proceedings of the 7th international conference on Privacy enhancing technologies*, pages 184–199, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against tor. In *WPES '07: Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20, New York, NY, USA, 2007. ACM.

[3] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security*

Symposium, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[4] M. Edman and P. Syverson. As-awareness in tor path selection. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 380–389, New York, NY, USA, 2009. ACM.

[5] N. S. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on tor using long paths. In *SSYM'09: Proceedings of the 18th conference on USENIX security symposium*, pages 33–50, Berkeley, CA, USA, 2009. USENIX Association.

[6] J. Feigenbaum, A. Johnson, and P. Syverson. Probabilistic analysis of onion routing in a black-box model. In *WPES '07: Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 1–10, New York, NY, USA, 2007. ACM.

[7] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding routing information. In *Proceedings of the First International Workshop on Information Hiding*, pages 137–150, London, UK, 1996. Springer-Verlag.

[8] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: estimating latency between arbitrary internet end hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, IMW '02, pages 5–18, New York, NY, USA, 2002. ACM.

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

[10] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia. A new cell counter based attack against tor. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 578–589, New York, NY, USA, 2009. ACM.

[11] J. McLachlan and N. Hopper. On the risks of serving whenever you surf: vulnerabilities in tor's blocking resistance design. In *WPES '09: Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 31–40, New York, NY, USA, 2009. ACM.

[12] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[13] G. O'Gorman and S. Blott. Large scale simulation of tor: modelling a global passive adversary. In *ASIAN'07: Proceedings of the 12th Asian computing science conference on Advances in computer science*, pages 48–54, Berlin, Heidelberg, 2007. Springer-Verlag.

[14] Onion-Info. Onion routing, 2006. `http://www.onion-router.net/`.

[15] V. Pappas, E. Athanasopoulos, S. Ioannidis, and E. P. Markatos. Compromising anonymity using packet spinning. In *Proceedings of the 11th Information Security Conference (ISC 2008)*, September 2008.

[16] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *SP '97: Proceedings of the 1997 IEEE Symposium on Security and Privacy*, page 44, Washington, DC, USA, 1997. IEEE Computer Society.

[17] Tor Project. FAQ, 2012. `https://www.torproject.org/docs/faq.html.en#EntryGuards`.

[18] C. Troncoso and G. Danezis. The bayesian traffic analysis of mix networks. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, pages 369–379. ACM, 2009.

[19] R. Wiangsripanawan, W. Susilo, and R. Safavi-Naini. Design principles for low latency anonymous network systems secure against timing attacks. In *ACSW '07: Proceedings of the fifth Australasian symposium on ACSW frontiers*, pages 183–191, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[20] M. K. Wright, M. Adler, B. N. Levine, and C. Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur.*, 7(4):489–522, 2004.

[21] M. K. Wright, M. Adler, B. N. Levine, and C. Shields. Passive-logging attacks against anonymous communications systems. *ACM Trans. Inf. Syst. Secur.*, 11(2):1–34, 2008.

[22] Y. Zhang. Effective attacks in the tor authentication protocol. In *NSS '09: Proceedings of the 2009 Third International Conference on Network and System Security*, pages 81–86, Washington, DC, USA, 2009. IEEE Computer Society.