WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Studying Vulnerability of TOR network for Website Fingerprinting Attacks

Ameya .A. Patil

ASM's Institute of Management & Computer Studies

ABSTRACT:

TOR is an open-source low latency anonymous network. Even though it is supposed to be an anonymous network, it still has its flaws. A passive attacker can record the user's anonymized web traffic activity and match it with pre-existing templates to decode it. A well-executed passive eavesdropping attack has an extremely high chance of a successful attack. An increase in delay and bandwidth overhead makes most of the proposals futile in real case scenarios. This paper discusses a method which can provide a defense against fingerprinting attacks without increasing the overhead and thus, can be used in the real world. It uses an adaptive padding approach to protect the privacy of TOR users. We shall discuss the shortcomings of the current approach and the advantages we can provide over it through adaptive padding..

Introduction:

TOR, which stands for The Onion Router, is a widely used anonymous network, whose objective is to provide anonymous browsing and thus promote internet privacy. TOR enables users to connect to the internet and browse, send messages anonymously, and has been historically used for both legal as well as illicit purposes. It is called The Onion Research because it uses layers of encryption in the application layer, like the layers of an onion. It is a volunteer-based network where anyone can let TOR use their system as a relay.

Website fingerprinting is a type of attack where an attacker can gather your browsing history. The attacker can record your packets and match them with pre-existing templates to find out which websites you are browsing. WF attackers can bypass HTTPS connections, SSH tunnels, one hop proxies and VPNS. WF attackers in recent times have achieved an accuracy rate of over 90% even against TOR and thus, break the anonymity of TOR users.

Most of the proposed defenses involve link padding, which adds dummy messages and delays to the packet flow to conceal messages. However, they come at the cost of latency and bandwidth overhead. Link padding is found to increase page loading time by a factor of two to four, while imposing bandwidth overhead by 40% to 350%. These methods have been proven to be too impractical to be used in real life scenarios. In this paper, we thus explore the design space of effective link-padding defenses with minimal latency overhead and modest bandwidth overhead.

Existing Technique

Most of the proposed designs are theoretical designs with no real-world implementation possible, and the one used by TOR is underwhelming.

Application-level defenses:

Application-level defenses. These defenses work at the application layer. HTTPOS modifies HTTP headers and injects HTTP requests strategically, while Randomized Pipelining, a WF countermeasure currently implemented in the Tor Browser, randomizes the pipeline of HTTP requests.

Super sequences and traffic morphing:

Recent works have proposed defenses based on generalizing web traffic traces. They create anonymity sets by clustering pages and morphing them to look like the centroid of their cluster. This approach aims to optimally reduce the amount of padding needed to confound the attacker's classifier.

Constant-rate padding defenses:

Dyer et al. evaluated the impact of padding individual packets [8], finding that this is not sufficient to hide coarse-grained features such as bursts in traffic or the total size and load time of the page. Dyer et al. simulated a proof-of-concept countermeasure called BuFLO, which used constant-rate traffic with fixed-size packets.

Problem Statement

Application-level defenses:
These are shown to be ineffective in stopping WF attacks.

Super sequences and traffic morphing:
These defenses, as well as traffic morphing techniques, have the shortcoming that require a database of webpage templates that needs to be frequently updated and would be costly to maintain.

Constant-rate padding defenses:
The authors report excessive bandwidth overheads in return for moderate security. The condition to stop the padding after the transmission ends is critical to adjust the trade-off between overheads and security. BuFLO stops when a page has finished loading and a minimum amount of time has passed, not covering the size of a page that lasts longer than the minimum time.
Tamaraw and CS-BuFLO, both attempt to optimize the original design of BuFLO. Instead of setting a minimum duration of padding, Tamaraw stops padding when the total number of transmitted bytes is a multiple of a certain parameter. This approach groups webpages in anonymity sets, with the amount of padding generated being dependent on the webpage's total size. Given the asymmetry of web browsing traffic, Cai et al. also suggest treating incoming and outgoing traffic independently, using different packet sizes and padding at different rates. Furthermore, the authors sketched CS-BuFLO as a practical version of BuFLO, extended with congestion sensitivity and rate adaptation. Following Tamaraw's grouping in anonymity sets by page size, they propose either padding up to a power of two, or to a multiple of the amount of transmitted application data. We question the viability of the BuFLO-based defenses for Tor. Their latency overheads are extremely high, such as two-to-three times longer than without defense, and the bandwidth overheads for BuFLO and CS-BuFLO are over 100%. In addition, due to the popularity of dynamic web content, it is challenging to determine when a page load completes, as needed in Tamaraw and CS-BuFLO. Nevertheless, in this paper, we compare our system against these defenses because they are the closest to meeting the deployment constraints of Tor.

Proposed Methodology

Adaptive Padding (AP) was proposed by Shmatikov and Wang to defend against end-to-end traffic analysis. Even though WF attacks are significantly different from these end-to-end attacks, AP can be adapted to protecting against WF due to its generality and flexibility. AP has the defender examine the outgoing traffic pattern and generate dummy messages in a targeted manner to disrupt distinctive features of the patterns — "statistically unlikely" delays between packets. Shmatikov and Wang showed that with 50% bandwidth overhead, the accuracy of end-to-end timing-based traffic analysis is significantly degraded. In the BuFLO family of defenses, the inter-arrival time between packets is fixed and application data is delayed, if needed, to fit the rigid schedule of constant packet timings. This adds delays in the common case that multiple real cells are sent all at once, making this family of defenses ill-suited for a system like Tor, as it would significantly harm user experience. By contrast, Adaptive Padding (AP) does not delay application data; rather, it sends it immediately. This minimal latency overhead makes AP a suitable candidate for Tor.

To clarify the notation adopted in this paper, we use outgoing to refer to the direction from the PT instance running at the client to the PT at the bridge, and conversely, incoming is the direction from the PT server to the client. The basic idea of AP is to match the gaps between data packets with a distribution of generic web traffic. If an unusually large gap is found in the current stream, AP adds padding in that gap to prevent long gaps from being a distinguishing feature. Shmatikov and Wang recognized the importance of bursts in web traffic and thus developed a dual-mode algorithm. In burst mode, the algorithm assumes there is a burst of real data and consequently waits for a longer period before sending any padding. In gap mode, the algorithm assumes there is a gap between bursts and consequently aims to add a fake burst of padding with short delays between packets. In this paper, we follow Shmatikov and Wang and define a burst in terms of bandwidth: a burst is a sequence of packets that has been sent in a short time period. Conversely, a gap is a sequence of packets that are spread over a long-time span.

WTF-PAD
We propose a generalization of AP called Website Traffic Fingerprinting Protection with Adaptive Defense (WTF-PAD). WTF-PAD includes implementation techniques for use in Tor and several link-padding primitives that enable more sophisticated padding strategies than the basic AP described above. These features include:

Receive histograms. A key feature to make padding realistic is to send padding messages as a response to messages received from the other end. In WTF-PAD, we implement this by keeping another AP state machine that reacts to messages received from the other PT endpoint.

Control messages. WTF-PAD implements control messages to command the PT server padding from the PT client. Using control messages, the client can send the distribution of the histograms to be used by the PT server. This way, the PT client is in full control of the padding scheme. It can account on received padding traffic and alert the user if relays in its circuits are sending unscheduled padding.

Beginning of transmission. Control messages can also be used to signal the beginning of the transmission. If we are in state S and a new page is requested, we will need to flag the server to start padding. Otherwise, the transmission from the first request to the following response is uncovered and

reveals the size of the index.html page.

Soft stopping condition. In contrast to Tamaraw and CS-BuFLO, WTFPAD does not require an explicit mechanism to conceal the total time of the transmission. At the end of the transmission, the padding is interrupted when we hit the infinity bin in the gap state and then the infinity bin in the burst state. The lack of a firm stop condition represents an advantage over existing link-padding-based defenses, which require a mechanism to flag the boundaries of the transmission. The probability of stopping will depend on the shape of the histograms at the end of the transmission.

Proposed algorithm

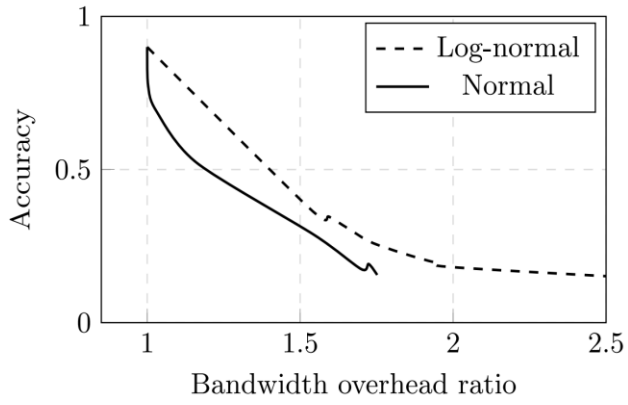
AP algorithm as a finite state machine as implemented in the PT client.

The AP algorithm is defined by two histograms of delays that we call HB (used in burst mode) and HG (used in gap mode). The histograms have a set of bins that span over the range of possible inter-arrival times. Each bin contains several tokens, which can be interpreted as the probability of selecting an inter-arrival time within the range of delays represented by that bin. The last bin, which we dub the "infinity bin", includes all possible values greater than the second-to-last bin.
AP implements the state machine shown in the above figure in each defense endpoint, i.e. both PT client and server. For simplicity, let us consider just the client's state machine in the following explanation. The operation of the server is symmetrical.

Burst Mode
As depicted in the diagram, AP starts idle (state S) until the packet with the HTTP request is pushed from the browser (R). This causes it to enter burst mode (state B), drawing a delay t from the HB histogram. Then it starts to count down until either new data is pushed or t expires. In the first case, the data is immediately forwarded, a new delay is sampled, and the process is repeated, i.e., it remains in burst mode. Otherwise, a dummy message (D) is sent to the other end and AP switches to state G (gap mode). The HB histogram is built using a large dataset of web traffic, out of which we sample the times between the end of a burst and the beginning of the following burst. Therefore, while we are in a burst, the delays we sample from HB will not expire until we find an inter-arrival time that is longer than typical within a burst, which will make the delay expire and trigger the G state

Gap Mode
While AP is in state G, it samples from histogram HG and sends dummy messages when the time it samples expires. The histogram for gap mode, HG, is built from a sample of inter-arrival times within a burst in traffic collected for a large sample of sites. That is, by sending packets with inter-arrival times drawn from HG, we can generate fake bursts that follow the timing distribution of an average burst. A transition from G back to B occurs upon either sampling a token from the infinity bin or receiving a real packet. Similarly, a transition from B to S happens when we sample a token from the infinity bin.
Note that AP immediately forwards all application data. Since sending a real packet means that the timeout expired, AP must correct the distribution by returning the token to its bin and removing a token from the bin representing the actual delay. This prevents the combined distribution of padding and real traffic from skewing towards short values and allows AP to adapt to the current transmission rate. If a bin runs out of tokens, to minimize its effect on the resulting distribution of inter-arrival times, we remove tokens from the next non-empty greater bin. In case all bins are empty, we refill the histogram with the initial sample.

Result



To evaluate the trade-off between bandwidth overhead and accuracy provided by WTF-PAD, we applied the attack on protected traces with different percentile values, ranging from 0.5 (low protection) to 0.01 (high protection) percentiles.
In the above figure, we show the trade-off curves for both normal and log-normal fits. We observe a steeper decrease in accuracy for the normal model with respect to the log-normal one. Remarkably, beyond a certain point (around 0.1 percentile), the tuning mechanism saturates to 15% accuracy for both models: percentiles lower than that point do not further reduce accuracy and only increase bandwidth overhead. The trend we observe is the cost in bandwidth exponentially growing with the protection level that the defense attempts to provide.

Conclusion

Inthis paper, we studied the vulnerability of current TOR network in protecting its users against WF attacks. and the limitations of the proposed methods as well as the areas where they fall short, thereby making TOR unusable if implemented. An alternative method was proposed and studied by us, which resulted in protection against WF attacks all the while resulting in very less bandwidth overhead.

References:

1. Ca i , X . , N ithyanand , R . , and Johnson , R . CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense.
2. Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, Mathew Wright: Toward an Efficient Website Fingerprinting Defense.
3. https://www.torproject.org/