Website Traffic Finger Printing Adaptive Defense (WTF-PAD): An Implementation

Privacy in the Internet Age - Professor Yixin Sun

Ishan Srivastava (yxs2qh) & Vikranth Nara (eyu8ps)

Fall 2024

Introduction

Website fingerprinting is a technique utilized by attackers to obtain extensive data from users without their consent. Attackers can accomplish this by browsing user's browsing habits and comparing the observed web traffic with pre-recorded templates. This process involves observing a user's traffic at key points, recording various features of network packets including the timing, direction, size, and sequence of packets, and matching this data with templates of similar patterns. This enables them to obtain a user's web activity without directly knowing what a user is viewing or doing online, bypassing the anonymity that comes with other methods including VPNs or Tor.

Unlike cookies, which are deletable by users, fingerprinting creates a consistent identifier that is very difficult to erase, enabling attackers to continuously track a user over time. Furthermore, fingerprinting collects data without storing it on a user's device, making it hard to detect and block. This is especially problematic as the data collected through fingerprinting can be utilized to build targeted advertising towards users, leading to unwanted and manipulative marketing techniques. In worst-case scenarios, malicious attackers can exploit fingerprinting to monitor user's online behavior, potentially leading to identity theft, fraud, and other cybercrimes.

There are current defense mechanisms that are utilized to combat fingerprinting, all of which are either limited in effectiveness or have performance issues. One such defense mechanism is randomizing HTTP requests. This involves altering the order and timing of HTTP requests to successfully break the patterns that fingerprinting relies on. However, this process degrades the performance of web applications and is not consistently effective against complex attacks. Another method involves traffic padding, including adding extra data to network packets to disguise the actual content. However, this method also increases the bandwidth and latency, making it highly impractical for everyday use.

For these various reasons, it is imperative for an effective solution to be implemented that successfully balances between robust protection and minimal performance impact. This includes accounting for excess bandwidth and noticeable latency. This report details one such solution that provides a reliable level of security against website fingerprinting in realistic conditions, such as well-known websites including CNN.

Methodology

The inspiration for the proposed solution obtained and detailed in this report originated from the "Toward an Efficient Website Fingerprinting Defense" paper. The researchers recognized the importance of creating a new solution for website fingerprinting that takes into account the efficiency and accuracy of blocking attackers by recommending an Adaptive Padding algorithm. In this algorithm, there are two main modes - burst mode and gap mode - that match the timing characteristics of web traffic and disguise different features used in user identification. The main idea involved adjusting the timing of packet transmission to blend in with web traffic patterns, preventing attackers from detecting periods of inactivity that leave a user vulnerable to fingerprinting.

The two modes occur based on the amount of packets sent in a given period. Specifically, burst mode occurs when there is a sequence of packets sent in a short period, while gap mode occurs when there are long periods of inactivity and no sequence of packets sent given a long period. Initially, the program will start in an idle state, waiting for any data packets to be transmitted. When a packet containing an HTTP request is detected, the program will transition to burst mode. In this mode, the program assumes that there is a burse of real data and high activity, causing it to track the inter-arrival times of the subsequent packets.

The program utilizes a burst mode histogram and a gap mode histogram to effectively model the distribution of inter-arrival times between packets, helping to ensure that the timing of dummy packets properly mimics realistic web traffic patterns. The burst mode histogram is built to provide the typical times between packets during a burst. Each bin in the histogram represents a range of possible delays, with tokens indicating the probability of selecting that delay given the specific range. On the other hand, the gap mode histogram consists of smaller delays that can be utilized to mimic a burst. This histogram is used when the algorithm is in gap mode and wants to simulate the inter-arrival times within a burst, helping to portray that the user is in burst mode instead of gap mode. In either mode, the algorithm selects a bin from the appropriate histogram, and then a delay from the bin range, uniformly at random which represents the time that the algorithm will wait before sending the next dummy packet. This ensures that packets aren't just sent randomly, which allows for increased variability in packet timings for more realistic traffic patterns. In burst mode, the dummy packets will be sent if the calculated delay expires without new data arriving, and the algorithm will transition to gap mode. In gap mode, the algorithm will send dummy packets at intervals determined by the gap histogram, in which the packers are timed to simulate typical burst patterns.

To further improve the efficacy of the web extension utilized to block fingerprinting, the sent dummy packets were all randomized by content and size. This helped to improve privacy as it ensured that each packet's hash was unique, making it much more challenging for attackers to recognize and filter out dummy packets based on predicted patterns. The final implementation of the program involved a web extension that had various essential features to ensure the user's privacy against website fingerprinting. To allow the user to toggle the extension on/off and adjust the padding intensity of the sent packets, our extension includes a simple, yet practical user interface to adjust personal settings and display performance metrics during the session. The performance metrics included a breakdown of dummy packets sent in each of the states (burst or gap) and the number of real packets intercepted. This feature helped quantify the efficacy of a user utilizing the extension and provided a real-time display of the metrics.

Results

The results that were found were expected as shown in the demo. The extensions page where the extension was loaded did not load any external servers except Google. The code shifted from idle to burst immediately as this new page opened, but then quickly shifted to the gap state. This is where the extension strived as it not only sent dummy packets to fill in the gap, but it randomized the size and content of the dummy packets to avoid showing a repetitive pattern for censors or servers that may block dummy packets. After this period of inactivity, a contrast was shown using an example of cnn.com, which was expected to transition back to a burst state and temporarily stop sending dummy packets. The video was cut here as it took a few minutes for the real packets from the user's end to CNN's server to stop sending as often. Once the initial high activity period of CNN servers cooled off, the extension switched back to the gap state and sent dummy packets as the histogram-based delay permitted. This was all shown neatly in the extension's user interface with clearly labeled metrics, showing total dummy packets sent and dummy packets sent in gap state and burst state, and even total packets intercepted.

Conclusion

The adaptive padding defense extension is a powerful solution for fighting website fingerprinting attacks. By shifting between burst and gap states while using histogram-based delay sampling, the extension obfuscates the user's traffic patterns. The size and content of dummy packets were randomized to prevent servers from identifying predictable padding signatures, which eventually leads to a unique website fingerprint for a user. The extension uses probabilistic sampling to ensure that delays between dummy packets are unpredictable and efficient against bandwidth. The extension offers users the ability to adjust padding intensity, allowing the user to have the ultimate choice of tradeoffs between high padding (more privacy but less usability) vs low padding (more usability but less privacy).

There are so many future possibilities with one approach being a machine learning model that could adapt its padding by learning from network traffic patterns. Instead of a static histogram-based approach that cannot adapt to real-time, unforeseen situations, a machine learning model could learn from each previous network interaction. Another approach is expanding protocol support across WebSocket, HTTP/2, QUIC, and others. This could transform the extension from a targeted solution into a comprehensive defense against advanced website fingerprinting attacks. Combining both approaches to develop an intelligent adaptive system that can learn from network traffic and stay a step ahead while being a universal solution that accepts support from most protocols is eventually the goal.

Works Cited

Juarez, M., Imani, M., Perry, M., Diaz, C., Wright, M. (2016). Toward an Efficient Website Fingerprinting Defense. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds) Computer Security – ESORICS 2016. ESORICS 2016. Lecture Notes in Computer Science(), vol 9878. Springer, Cham. https://doi.org/10.1007/978-3-319-45744-4_2