Fingerprinting Attack on the Tor Anonymity System

Yi Shi and Kanta Matsuura

The University of Tokyo, Japan
{shiyi,kanta}@iis.u-tokyo.ac.jp

Abstract. We present a novel way to implement a fingerprinting attack against Onion Routing anonymity systems such as Tor. Our attack is a realistic threat in the sense that it can be mounted by a single controller of entrance routers and furthermore require very few resources. The conventional fingerprinting attack based on incoming traffic does not work straightforwardly against Tor due to its multiplex and quantized nature of traffic. By contrast, our novel attack can degrade Tor's anonymity by a metric based on both incoming and outgoing packets. In addition, our method keeps the fingerprinting attack's advantage of being realistic in terms of the few required resources. Regarding evaluation, the effectiveness of our method is discussed in a comprehensive manner: experimentally and theoretically. In order to enhance further studies and show the significance of our idea, we also discuss methods for defending against our attack and other applications of our idea.

Keywords: fingerprinting attack, anonymity system, Tor, onion routing.

1 Introduction

The internet brings us convenience, but also hurts our anonymity. With some tools, it is easy for an attacker to eavesdrop activities of other users. Individuals and organizations need anonymity on the Internet. People want to surf webpages, make online purchases, and send email without exposing their identities and activity patterns to others. Encryption solves some parts of this problem, but not everything. It can hide the communicating contents, but can do nothing with the packet headers, which reflects the identity of communication parties. Anonymity systems provides the foundation for users to share information over public networks without compromising their privacy.

A simple example: the websites nowadays keep profiles of users to provide a more suitable services. Large-scale B2C sites like Amazon will supply more suitable items for each user based on their surfing history and transaction records. If we bought some game software, then some games with the same platform will be recommended to us the next time. It makes seller provide better service and gives the buyer convenience, but it also hurts our privacy, since our transaction records could also be misused by the seller.

© Springer-Verlag Berlin Heidelberg 2009

Anonymity systems could keep websites from profiling individual users. Anonymity systems could also be used for socially sensitive communication: forums or chat rooms for survivors of serious criminal cases, or people with specific illnesses. Journalists could use anonymity system to communicate with whistle-blowers and dissidents safely. Corporations could use anonymity system as a safe way to conduct competitive analysis.

Moreover, big organizations such as embassies use anonymity systems to connect with their overseas headquarters. Law enforcement uses it for collecting evidence without alerting suspects. Non-governmental organizations usually use anonymity systems to connect to their website while they are abroad, without notifying everybody nearby what they are working with.

Modern anonymity systems was first introduced by Chaum as early as 1981 [4]. Many discussions followed that, both theoretical and practical, such as The Dining Cryptographers Problem [3], Babel [8], Crowds [13], Tarzan [7], MorphMix [14] etc. Among all of these anonymity systems, Tor [6] is one of the most widely used. It employs the Onion Routing scheme to anonymize the traffic, by using layered cryptography combined with widely distributed relay nodes in different administrative domains.

Although Tor is a well developed and still improving state-of-the-art anonymity system, it is vulnerable to traffic analysis attacks just like other low-latency anonymity system. Many researches concern about how to reveal the relationship between users who are using the anonymity system and how to degrade the anonymity. We believe that research on attacks against anonymity system will help us to understand the concept of anonymity more clearly, and help the anonymity system become more secure in the future.

The remaining parts are organized as follows: we will summarize the related works in Section 2, the attack plan in Section 3, and the experiments in Section 4. Countermeasures will be discussed in Section 5, and finally we will give the conclusion with some open questions in Section 6.

2 Related Works

After the early-day discussions about theoretical anonymity systems, many practical anonymity systems were presented in recent years as we discussed in Section 1. Practical anonymity systems trade off resistance against some kinds of adversaries with lower latency and better performance to provide usability to attract users. The mainstream of attacks toward anonymity systems are end-to-end confirmation attacks, which is mainly based on timing-based analysis techniques. It is widely researched and many papers exist, like [1,10]. It is a really powerful attack but also requires a strong assumption that the adversary controls both of the nodes adjacent to communication partners.

In low-latency anonymity system design, defense against timing attacks is merely considered as an desirable feature to achieve. Like in [6], Tor's designer claimed that known solutions seem to require either a prohibitive degree of traffic padding or an unacceptable degree of latency. Both will greatly decrease the usability of the anonymity system.

In 2008, Pries et al. presented a novel confirmation attack [12]. They do not use the classical way to analyze whether the two flows observed by two compromised nodes are identical, but just copy a packet from the flow by the entry node, and then choose the right time to replay it. The replayed packet could go through the whole path. And in the exit node, it will cause an integrity error, which is an unusual event. If the adversary could observe this event, he could assume that the two nodes are in the same path.

Although end-to-end confirmation attack dominates the attack research in this area, the disadvantage is also very clear - it is not practical enough. The assumption about control of both entry and exit node is unrealistic in the real world. Generally, we cannot be a global eavesdropper. The nodes in anonymity systems are usually distributed among the whole world. Even for big organization, such as governments or ISPs, it is also hard to control both ends of a path.

There are also other types of attacks towards anonymity systems. Fingerprinting attack was raised relatively early by Hintz in 2003 [9]. It uses different file sizes in a webpage to make a fingerprint, and try to use it to identify user's latter actions. It was first designed against SafeWeb. We will discuss it in detail later.

Chakravarty et al. presented another interesting idea in [2], 2008. In their threat model, the adversary needs not to be any point along the path. A completely separated attacker just observes all the bandwidths of nodes in the anonymity system using Linkwidth, a bandwidth-estimation technique. If several nodes's bandwidth changed by the same amount, then we could assume these nodes are in the same path. The disadvantage of this attack is that it assumed that the adversary occupies sufficient bandwidth and stands at a "vantage" point, which means that the bottleneck in the path connecting the adversary to the victim relay should always be the latter.

In [5,11], a well-known class of attacks called statistical disclosure treat the whole system as a black box. It correlates traffic that enters and exits it to discover some communication patterns. But the used model of anonymity system is somewhat simple and theoretical. The same as timing attacks, the threat model they used is out of anonymity system's consideration.

3 Fingerprinting Attack on Tor

In this section, we will first review the original fingerprinting attack, discuss the characteristic of Tor, and then raise our proposal of the fingerprinting attack on Tor.

3.1 The Original Fingerprinting Attack

Generally, when a user visits a typical webpage, it consists of many different files. First, the HTML file is downloaded from the site, after the links contained in the HTML file is analyzed by the browser and pictures included in the page, background music, flv movie, etc. would also be downloaded after that. If we surf the webpage at www.yahoo.co.jp, about 23 files would be retrieved from the server. Each of files has a specific file size.

In a typical browser, such as Microsoft Internet Explorer, each file would be downloaded via a separate TCP connection. So we can easily detect every TCP flows since they use different ports to transfer the files. Then, the attacker can determine the size of each file being returned to the client. All the attacker needs to do is just count the total size of the packets on each port.

This kind of attack can not only be applied to the plain flows, but also to the simple anonymity system like SafeWeb. Because common encryption methods do not try to obfuscate the transmitted data, for both performance and requirement reasons. If someone monitors the Safeweb user, the number and approximated file size could be determined. For example, the eavesdropper found that the user created 3 connections with the same target, each of the connections received respectively 1324 bytes, 582 bytes, and 32787 bytes. So each of these transfer sizes corresponds to a certain file directly, and the *fingerprint* of a webpage consists of the set of file sizes.

So the attacker could first try to build the fingerprint of the file sizes of webpages, then monitor the user. When the user is surfing a webpage, connections and related data could be detected by the attacker. Then the attacker just compare the connect data with a set of fingerprints, choose the closest one, then "guess" that the page is what user surfing now. The attack is low-cost, easy to apply, and really hurts the user's anonymity.

3.2 The Characteristics of Tor

Tor is a low-latency, well developed anonymity system. It uses multi-hop encrypted connections to protect sender and/or receiver anonymity. Tor extends the former onion routing scheme by adding some features like integrity protection, congestion control, and location-hidden service. Tor can be used for both sender and receiver anonymity. Sender anonymity could help a user to use services without disclosing their identities. In Tor's design, it employs two significant characteristics, which prevents the fingerprinting attack to some extent.

First, Tor employs quantized data cells, each data cell is fixed at 512 bytes. So it is obviously difficult for an attacker to detect the accurate size of files transferred by separated connection stream.

Second, Tor uses multiplexing to combine all the TCP streams into one connection. This is not for the safe aspect at first. The original Onion Routing creates a path for each TCP stream. But for the expensive communication cost, Tor decides to use multiplexing to reduce the expensive path-establish cost. And it also provides some resistance to the client against fingerprinting attacks, for the attacker cannot distinguish the connections between each other easily.

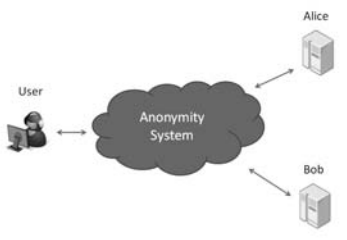
3.3 Threat Model of Fingerprinting Attack

Although many attacks toward low-latency anonymity systems are successful in their assumed environment, Tor and other anonymity systems are considered to be secure in practical use. Many attacks involve a strong adversary, who could perform end-to-end confirmation or even global eavesdrop. And in practical world, it is obviously difficult to achieve this kind of requirement. Even for big organizations to observe all the nodes distributed in the whole world is almost impossible. The advantage of fingerprinting attacks is the low resource requirement. The adversary only needs to occupy the entry point of the user. Compare to the end-to-end confirmation attacks, they just use the resources which much easier to satisfy, make it more possible to implement.

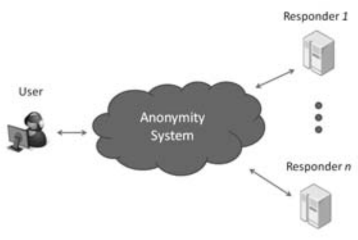
Our fingerprinting attack on Tor uses the same threat model with the fingerprinting attack by Hintz, the attacker is assumed to occupy the entry router of the user and observe all the data flows from the user. He wants to guess what webpage the user is surfing now. The design objective of Tor is attempting to defend against external observers who cannot observe both sides of a user's connections. So we think our threat model is appropriate against low-latency anonymity system.

Let us describe the model more formally, assume there is a user and two responders: Alice and Bob. An adversary can watch all the connections related to the user. First, the adversary could use the anonymity system to visit Alice and Bob for many times. Then the user visits either Alice or Bob using the anonymity system under the adversary's observation. Then the adversary would guess which responder the user connected to. We have some a priori probability, which models our suspicion about who is communicating with whom. More precisely, the a priori probability that the user is communicating with Alice is p and the a priori probability that user is communicating with Bob is $1 - p$. If we have no priori information, $p = 1/2$. See Figure 1(a).

Then, the model could also easily be extended to n responders, assume now there are n responders, from Responder 1 to Responder n. First, the adversary could use the anonymity system to visit any responder for many times. Then the user visits one responder using the anonymity system under the adversary's observation. Then the adversary would guess which responder the user connected to. We have some a priori probability, which models our suspicion about who is communicating with whom. More precisely, the a priori probability that user is communicating with Responder i is p and the a priori probability that the user is communicating with other responders are $1 - p$. If we have no priori information, $p = 1/n$. See Figure 1(b).



(a) Model with 2 responders



(b) Model with n responders

Fig. 1. Model in Fingerprinting Attack

3.4 Fingerprinting Attack on Tor

So we come to make our fingerprinting attack towards Tor. The biggest problem is that the only connection makes it hard for the adversary to distinguish each file size and the characteristic of the webpages becomes hard to define.

Generally, if we observe the traffic flow from/to the user, we will see a sequence of packets. If we use the outflow from user to separate the flow, we will see some interesting things. Some intervals may be very short, like 1 or 2 packets between two outflow packets. That means this interval transferred some small files or does some protocol transactions, etc. And some intervals may be relatively long, like 5 or 6 packets. This means a bigger file is being transferred. And after the TCP sliding window is fulfilled, the user send the acknowledge packet and continue the transfer process. If the network condition remains stable, this traffic pattern will not change much. So, for the webpages with different files and different loading process, we can distinguish them to some extent.

With a specific packet sequence, we could use the method described above to make a continuous intervals with the different number of packets. We call *all the inflow packets in a sequence, without any outflow packet placed in them*, an **interval**. We then define a vector $\boldsymbol{V} = (v_1, v_2, \ldots, v_n)$, where v_i means "the number of intervals with i packets". $n_{\boldsymbol{V}}$ means "the total number of intervals in $\boldsymbol{V}$". We build a fingerprint vector $\boldsymbol{F}$ in advance. Let weight w defined as $n_{\boldsymbol{V}}/n_{\boldsymbol{F}}$ or $n_{\boldsymbol{F}}/n_{\boldsymbol{V}}$ which is smaller (equal when $n_{\boldsymbol{V}} = n_{\boldsymbol{F}}$) than 1. So we use this formula to calculate the similarity S:

$$S = \frac{\boldsymbol{V} \cdot \boldsymbol{F}}{\|\boldsymbol{V}\|\|\boldsymbol{F}\|} \cdot w \tag{1}$$

If we have several fingerprints, we could calculate observed $\boldsymbol{V}$ with each $\boldsymbol{F}_i$ to get several similarity S_i, then we could sort all the S_i and make the assumption the user is surfing the webpage with the $\boldsymbol{F}$ correlated to the largest S_i.

In the ordinary fingerprinting attack, because a webpage is usually consists from about 20 to 30 files, and each file has its own unique file size. It means that the number of distinguishable webpages is very large. But in our work, the information we used is really limited due to the multiplexing. So if the number of fingerprint we use is too large, we may not have a very high detection rate. When the webpages the user may access are too many, after sorting the similarity S, we do not make the assumption only with the biggest S, but also using a threshold value θ instead. All fingerprints with calculated score larger than θ could be the possible page the user has seen. And we could make this as a set. If we could make sure the user is surfing the same page again and again (but we do not know which page he is watching), then we get other sets. Combine these sets and finally we could get the most possible answer.

3.5 The Choice of Fingerprints

So far we have discussed our threat model, the score formula and the method to recognize the page. But how can we choose a fingerprint?

Generally, any vector $\boldsymbol{V}$ could be a fingerprint but the unique noises are also included in the fingerprint. An adversary may do the sampling work in advance and make a lot of vectors from one page. He wants to use them to achieve a higher detection rate from the data, so which one should he choose?

The fingerprint choosing method is also discussed in ordinary fingerprinting attack paper: the author claims that we should choose the smallest sizes sampled for each file. It is an intuitive idea that if we observed the same thing with the smallest size, then it must be with minimum noises. But in our opinion, for the adversary has almost same network condition as the user. The fingerprint should not only reflect the characteristic of webpage, but also the network condition of user.

We could assume the attacker access a webpage n times and recorded vectors as $\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_n$. We calculate the scores with each other by formula 1. Then we could get the scores S_{ij} calculated from $\boldsymbol{V}_i$ and $\boldsymbol{V}_j$ $(i = 1, 2, \ldots, n-1, j > i)$. So we could choose $\boldsymbol{V}_i$ with the maximum S'_i as the fingerprint vector $\boldsymbol{F}$, which represents:

$$S'_i = \prod_{j}^{j \neq i} S_{ij} \tag{2}$$

4 Evaluation

4.1 Environment and Data Collecting Method

We use Windump to capture the Tor packets (Version 0.2.0.34) on a PC with Intel Core2 Duo 1.86G, 4G RAM, Vista Business. We shall run the windump to observe the port 9001 on the host machine. Then we use firefox which installed TorButton to surf the webpage. After a webpage is fully loaded, we stop capturing the packets. We use Wireshark to open the pcap file, filter the obvious noise manually. More precisely, in a short period, all the connections raised from Tor are going through the same path. So most of the packets will obviously have the same destination address (Actually, this address refers to the first node in the path). And some packets with other destination addresses refer to other control packets used in Tor, like establishing new paths. After this process, a data is recorded. We also wrote some programs to analysis the captured data to make the calculation.

4.2 Data Analysis

First, we shall use Alexa Ranking - Top Sites in Japan[1] to see how our method works in a practical environment. In Figure 2, we use n to represent the top n sites' mainpages we used to implement the experiment. We choose the top 20 sites to implement the experiments.

In the experiment, we choose top $n = 5, 10, 15, 20$ sites, and built fingerprint of the site. Then we surfed webpages and recorded the user activity vector, compared with the fingerprint, and guessed which website user is surfing. The success rate represents in the Figure 2.

[1] http://www.alexa.com/topsites/countries/JP

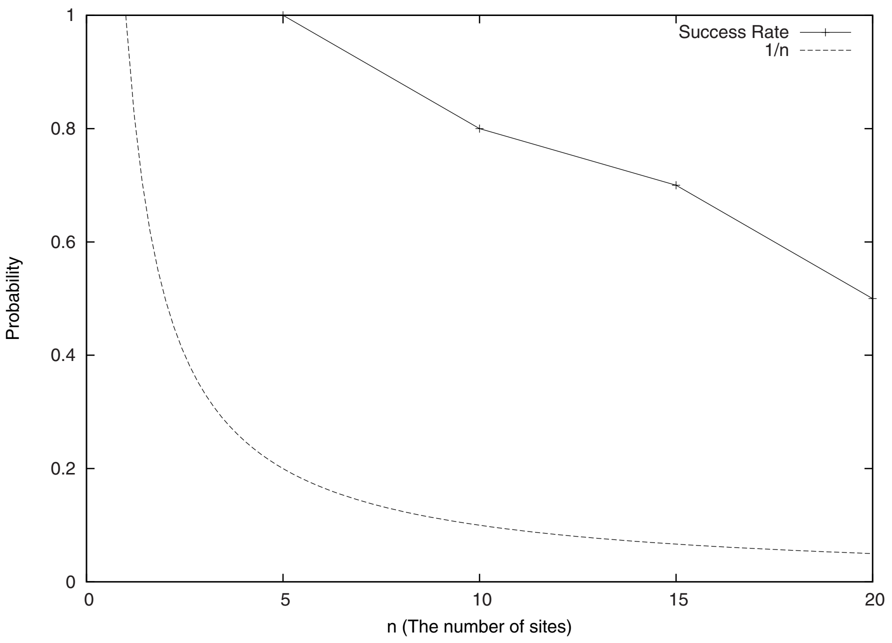


Fig. 2. Success Rate in Different n

From Figure 2, we could see that: the success rate is relatively high when n is small. With n increases, the success rate decreases significantly. There are several reasons for this: First, the information we used is limited, the fingerprint of the webpage is not so unique. So obviously the success rate decreases when n increases. Second, some pages are not suited for fingerprinting, like youtube[2], amazon[3]. The items on these sites would change from time to time, which hurts the consistency of fingerprinting. Other sites like Yahoo[4] will have ads change frequently, too. But compared with other parts of the page, the ratio of ads is not so large and we could just treat them as noise. Third, the pages we have chosen are all homepages, with the similar design, it increases the difficulty of distinguishing. Fourth, the noise in practical network affects the result a lot, and that's why we need to implement our method instead of just making the simulation. Last, there are some sites hard to see the difference but still be counted as different ones, like Google[5] and Google Japan[6]. This problem also exists in the distinguishing between the original page and phishing page. We will discuss the success rate in more formal way in the following section.

[2] http://www.youtube.com/
[3] http://www.amazon.co.jp/
[4] http://www.yahoo.co.jp/
[5] http://www.google.com/
[6] http://www.google.co.jp/

4.3 Theoretical Discussion

In this part, we will discuss the effectiveness of this attack in theory. We will discuss two topics: The factors that related to success rate and make an estimate of how many webpages (or webpage groups) could be distinguished without too high error rate.

First, Let us discuss with the success rate. We will use the method in [10] to show the attack success probability formally: We use $V \sim F$, to indicate that the attacker's test says that vector $\boldsymbol{V}$ and fingerprint $\boldsymbol{F}$ are from the same site. And we use $V = F$ to indicate that the event that vector $\boldsymbol{V}$ and fingerprint $\boldsymbol{F}$ are from the same site. We have the *false positive rate*, $Pr_{fp} = Pr(V \sim F|V \neq F)$, and *false negative rate*, $Pr_{fn} = Pr(V \nsim F|V = F)$, are both known. We can therefore obtain:

$$\begin{aligned} Pr(V \sim F) &= Pr(V \sim F|V = F)Pr(V = F) + Pr(V \sim F|V \neq F)Pr(V \neq F) \\ &= (1 - Pr_{fn})Pr(V = F) + Pr_{fp}(1 - Pr(V = F)) \\ &= (1 - Pr_{fn} - Pr_{fp})Pr(V = F) + Pr_{fp} \end{aligned}$$

Which leads us to obtain:

$$\begin{aligned} Pr(V = F|V \sim F) &= \frac{Pr(V = F \wedge V \sim F)}{Pr(V \sim F)} \\ &= \frac{Pr(V \sim F|V = F)Pr(V = F)}{Pr(V \sim F)} \\ &= \frac{(1 - Pr_{fn})Pr(V = F)}{(1 - Pr_{fn} - Pr_{fp})Pr(V = F) + Pr_{fp}} \end{aligned} \tag{3}$$

Suppose $Pr(V = F) = 1/n$, e.g., we are observe n sites and the adversary has no additional information about which site the user is likely surfing. Then, the success probability depends on Pr_{fp} and Pr_{fn}.

In the simplest case, we first assume the false positive rate and false negative rate are constant. Then, with $Pr_{fn} = Pr_{fp} = 0.1$ and $n = 10$, which means the user could surf 10 webpages and we've made all the fingerprints of them, we could get $Pr(V = F|V \sim F) = (0.9 \cdot 0.1)/(0.8 \cdot 0.1 + 0.1) = 50\%$. And if we improve Pr_{fn} and Pr_{fp} to 0.01, then with 10 webpages, the success probability is about 91.7%. As n rises to 100 webpages, this probability also falls to only 50%. With $n = 1000$, it is less than 10%.

But as we see in the evaluation above, the Pr_{fn} and Pr_{fp} rises with n. So, we will describe the false positive rate and the false negative rate as a function of n. We also use the assumption $Pr(V = F) = 1/n$ discussed above. Then the Equation 3 would be:

$$\begin{aligned} Pr_{Success} &= \frac{(1 - F_{fn}(n))/n}{((1 - F_{fn}(n) - F_{fp}(n))/n) + F_{fp}(n)} \\ &= \frac{1 - F_{fn}(n)}{1 - F_{fn}(n) - F_{fp}(n) + F_{fp}(n) \cdot n} \end{aligned} \tag{4}$$

Actually, it is almost impossible to make a reasonable function to reflect the relationship between n and the error rate, for it is affected greatly by the sites we have chosen. But we could assume Pr_{fn} and Pr_{fp} have a linear relationship with the increase of n, then from the Equation 4, we could see the numerator falls with n, and the denominator increases even faster, which will leads the success probability decreasing even faster.

The equations we listed above tell us if we want to increase the success rate, there are several points: First, to improve the accuracy, that is, decrease the false positive and false negative rate. Second, make the webpages we need to guess as few as possible, what means make the n lower. What's more, we assume the adversary knows nothing in advance. So the $Pr(V = F)$ equals $1/n$. But if in some situation, $Pr(V = F)$ is greater than $1/n$, which means the adversary gets some additional information from other ways, the success rate itself will also be raised.

Then, we shall come to how many webpages we could distinguish without high error rate, if not choose the webpages randomly but we could choose by ourselves.

Notice that the similarity S consists of two components, the relative interval ratio and the vector's dot product. First, we take a look at the relative interval ratio. We have implemented an experiment to get that the mainpage of Yahoo Japan have an average interval of 159.2105, with the standard deviation of 14.8495. Figure 3 shows the distribution of intervals of Yahoo Japan.

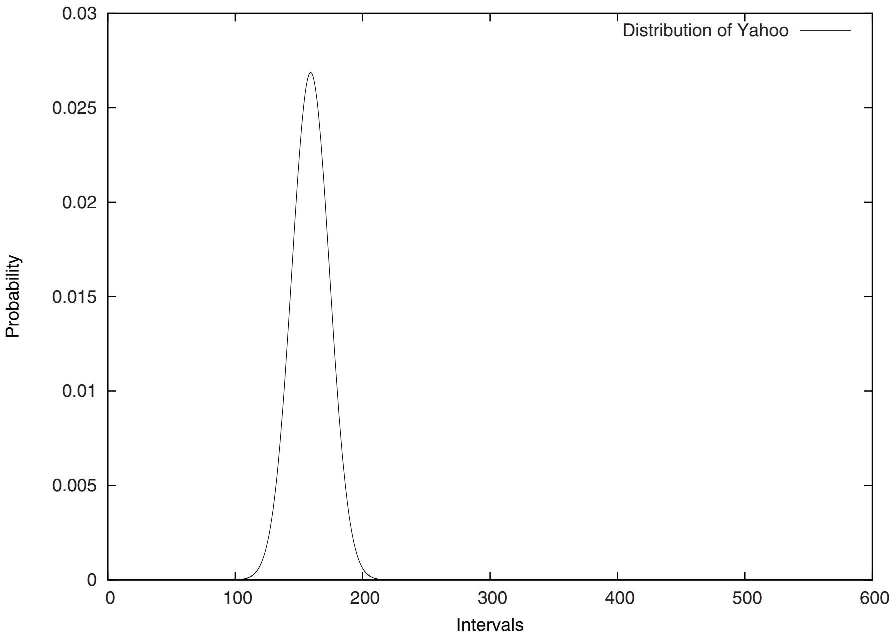


Fig. 3. Distribution of Intervals of Yahoo Japan

From our observation, the intervals of webpages often fall in the range from 50 to 600. We can choose the page freely here, webpages with more than 1000 intervals are not so rare in practical. But here we just want to make an theoretical estimate; we will choose the range of interval up to 600.

As our experiment about Yahoo Japan, the standard deviation is approximately 10% of the intevals, that means, with about ± 20% gap between two sites, there is about 95% chance the vector could be recognized correctly. Roughly speaking, there are $log_{1.4}(600/50)+1 \approx 8.38$ slots for us to choose webpages with high detection rate.

Then we come to the dot product of vectors. In our implementation, the vector is limited to 5-dimension. Because intervals with more than 5 packets are so rare, intervals with more than 5 packets would be treated as one with just 5 packets.

Theoretically speaking, if we use 20% gap as we do in the discussion about interval, then there are a lot of available slots for us to choose, considering we have 5-dimension to do the permutation. But actually, in typical situation, the intervals with 1 or 2 packets dominated in the total dimensions, for there are a lot of transactions to be done(Also, in some extreme condition, such as file transferring, we could expect to observe a lot of long intervals). By our observation, in the situation with similar intervals, there are about 3 or 4 significantly different results. Combine this with the result about interval, we have approximately 20 to 40 available slots for choosing webpages to be recognized.

It is hard to improve this result, unless we could find some way to significantly reduce the noise. But it is expected to be improved by following research. For example, our method does not employ with time/latency, if we could employ it into the calculation of score, the number of pages we could detect maybe greatly increased.

5 Countermeasures and Discussions

In this section, we will discuss some countermeasures to our attack and some countermeasures which is believed to be effective toward fingerprinting attack and its possible extensions. And there are also some open questions waiting to be further researched.

5.1 Change the Fixed Cell Size

It is believed a longer Tor cell size will make it harder to attack, e.g. Increase the Tor's cell size from 512 bytes to 1024 bytes. But unfortunately, in our attack scheme, it will have little impact. Tor's fixed cell size gives the system some advantage in traffic analysis theoretically. But the protocol it uses is still built on TCP. So no matter what the cell size is, it could still be wrapped by TCP packet and be divided into 1500 bytes a packet in ethernet. If there exists a scheme to analysis Tor's cell from TCP packets, this defense method could have some results, but not in our proposal.

5.2 Make Odd Requests

Odd requests refers to some surfing actions which is unusual. For example, always surfing several pages meanwhile, restricting the scripts or pictures downloading, etc. If there is a page with vector $\boldsymbol{V}_1$ and another with vector $\boldsymbol{V}_2$, then when we view these two pages at same time, the adversary could get a $\boldsymbol{V}_3$ equals $\boldsymbol{V}_1+\boldsymbol{V}_2$, and $\boldsymbol{V}_3$ has no difference with the vector $\boldsymbol{V'}_3$ which has the same elements as $\boldsymbol{V}_3$, although it may be observed from one single page. Other odd requests like the restriction on downloading some specific files. Like the combination of two pages, it is also difficult for an adversary to match the characteristic from the fingerprint vector. Although this kind of defensive method seems to be so effective against fingerprinting attack, it depends on the user's action. But we cannot make the system's security depends how users use this system. It is dangerous to assume the users have the knowledge in security and will work in a secure way. Moreover, it is not hard to develop some kind of explorer plug-in to achieve this objective. Like TorButton, if we activate this plug-in, it will randomly disable some kinds of files in the current webpage, maybe forbid running script or download pictures. It will help us in the anonymity, but we do not think users would really accept some plug-ins like this.

5.3 Run Own Entry Node

Entry nodes are also called "guard nodes". And people believed that they could guard your traffic from malicious nodes. First, it is not so useful to run a node by oneself when the adversary occupies the entry router. Especially the time when they are allocated in the same ethernet. Second, to run an own entry node and achieve the requirement of anonymity is very costly. That means, to make an adversary unable to distinguish the flows from a user. The own node may accept many connections from other users, which may hurt the usability of company's network and unacceptable. But running a node with only permitted user also makes this node meaningless. How to make the balance could be a question to network administrators.

5.4 Defensive Dropping

Defensive Dropping is a defensive method against timing attacks introduced by Levine et al. [10]. It employs the mechanism of dummy packets. The communication initiator constructs some of the dummy packets. These dummy packets are transferred on the path as normal packets. But to each packet, there is a probability P_{drop} to be dropped in each node rather than passing it on to the next node. If the number of dummy packets is randomly placed with a sufficiently large frequency, the correlation between every visiting will be greatly reduced. As we see in this theoretical discussion part, the increasing in the false positive rate and false negative rate will greatly reflected in the situation where we need to recognize object from a lot of webpages.

Although it is an effective way to defend against not only end-to-end attacks but also fingerprinting attack, we must notice that it is a really expensive defense mechanism, especially in low-latency anonymity system. If the number of the dummy packets is relatively small, then these dummy packets are no more than normal background traffics. But with many dummy packets, it is unacceptable for consuming so many resources. What's more, use more dummy packets in sensitive connection is also not a good idea for it gives the adversary a clear sign to notice the sensitive data transfer. So how to determine the sufficient number of packets will leave to be an open question for further research.

5.5 Other Applicable Situations

Although this attack is mainly designed towards Tor, it could also be applied in other situations.

First, it is not hard to see every anonymity systems with multiplexing or quantized cells could be attacked by our proposal. And even without multiplexing or quantizing, our proposal also works. You can treat all the connections as if they were one. But it would be ineffective for we discard some useful information by this process.

Second, this kind of attack can not only be applied attacking information regarding webpage surfing, but also other forms of network activities. For example, in instant chatting, there should be differences between one who talks quickly but every sentence is short and another merely talks but using long paragraphs. This kind of differences could be reflected in their traffic flows, although the significance may not be high enough to be detected.

What's more, our scheme do not only apply to the entry point of the path, but also the exit point. Imaging that if you are a curious server administrator who is running a system which accepts both anonymous and non-anonymous visits from anonymity systems. You could record the patterns when users visiting your sites in non-anonymous mode. And someday, for some purpose, a user visits your sites anonymously. Then you could use this scheme to guess which user it is. Just by comparing the historical patterns and the flows you observed.

We just simply described some other possible situations for the application of our proposal. Theoretically, for any kinds of activities with stable traffic patterns, our proposal could be a potential threat. Although the effectiveness of our proposal still needs to be improved by further research.

6 Conclusion

In this paper, we have discussed a novel fingerprint attack against the Tor anonymity system. Our scheme works by analyzing users' traffic flows in the anonymity system. We use outflow packets to divide a flow into several intervals, turn a flow into a vector, and give a formula to calculate the similarity of two vectors in this scheme. It can easily be implemented by network administrators, governments, or ISPs. The experimental results showed our scheme to be very effective. The user's anonymity is really degraded by this simple and practical attack. As we have discussed, this effectiveness has a potential of being improved even more. Also, we have given a theoretical reasonable estimate of the effectiveness, showed the simple model of fingerprinting attacks on anonymity systems. Moreover, the result showed the need for the use of dummy traffic in the low-latency anonymity systems, but how to evaluate it is still left as an open question. As future work, we may improve the attack method, especially the scalability. We could also try to present some new threat models.

References

1. Back, A., Möller, U., Stiglic, A.: Traffic analysis attacks and trade-offs in anonymity providing systems. In: Moskowitz, I.S. (ed.) IH 2001. LNCS, vol. 2137, pp. 245–257. Springer, Heidelberg (2001)
2. Chakravarty, S., Stavrou, A., Keromytis, A.D.: Identifying proxy nodes in a tor anonymization circuit. In: Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems, Washington, DC, USA, pp. 633–639. IEEE Computer Society, Los Alamitos (2008)
3. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology 1, 65–75 (1988)
4. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–90 (1981)
5. Danezis, G.: Statistical disclosure attacks: Traffic confirmation in open environments. In: Security and Privacy in the Age of Uncertainty, International Conference on Information Security (SEC 2003), Athens, Greece, May 26-28, pp. 421–426. Kluwer, Dordrecht (2003)
6. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium (August 2004)
7. Freedman, M.J., Morris, R.: Tarzan: a peer-to-peer anonymizing network layer. In: Proceedings of the 9th ACM conference on Computer and Communications Security, pp. 193–206. ACM, New York (2002)
8. Gülcü, C., Tsudik, G.: Mixing E-mail with Babel. In: Proceedings of the Network and Distributed Security Symposium - NDSS 1996, pp. 2–16. IEEE, Los Alamitos (1996)
9. Hintz, A.: Fingerprinting websites using traffic analysis. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 171–178. Springer, Heidelberg (2003)
10. Levine, B.N., Reiter, M.K., Wang, C., Wright, M.K.: Timing attacks in low-latency mix-based systems. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 251–265. Springer, Heidelberg (2004)
11. Mathewson, N., Dingledine, R.: Practical traffic analysis: Extending and resisting statistical disclosure. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 17–34. Springer, Heidelberg (2005)
12. Pries, R., Yu, W., Fu, X., Zhao, W.: A new replay attack against anonymous communication networks. In: Proceedings of the IEEE International Conference on Communications, ICC 2008, May 2008, pp. 1578–1582 (2008)
13. Reiter, M., Rubin, A.: Crowds: Anonymity for web transactions. ACM Transactions on Information and System Security 1(1) (June 1998)
14. Rennhard, M., Plattner, B.: Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In: Proceedings of the Workshop on Privacy in the Electronic Society, Washington, DC, USA (November 2002)