Study on OS Fingerprinting and NAT/Tethering based on DNS Log Analysis

Deliang Chang
Tsinghua University
chdlgs@gmail.com

Qianli Zhang
CERNET Center
Tsinghua University
zhang@cernet.edu.cn

Xing Li
CERNET Center
Tsinghua University
xing@cernet.edu.cn

ABSTRACT

OS fingerprinting and NAT detection are considered important in various researches like network troubleshooting, deployment of services. Previous passive approaches usually require raw network traffic, which is often difficult to deploy. In this paper, a novel method is designed to fingerprint the OS and classify the NAT only using DNS log. Features of the Windows, MacOS/iOs, Android and Linux operating systems can be automatically extracted from labelled DNS log. With these features a simple classifier can fingerprint the OS types of these devices accurately. We apply this algorithm on data set from a large scale network. Analysis also reveals that nowadays the Windows operating systems are widely used in tethering or NAT, which is contrary to our previous knowledge.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network OperationsNetwork Monitoring

General Terms

Algorithm, Measurement

Keywords

OS Fingerprinting, DNS, Supervised Learning

1. INTRODUCTION

Fingerprinting operation system (OS) on a remote device passively is often required by network administrators to make network policy or deploy network services. OS fingerprinting is also widely used in NAT detection.

There are some previous works about OS fingerprinting, since it's a research area that attracts high attention for a long time. Some approaches [1, 7, 3] are active, which means sending probe packets to devices need to be fingerprinted. Another approaches [12, 8, 6] uses passive ways. They fingerprint the OS by processing information collected from those devices passively. They used some fields in TCP/IP headers or other kinds of traffic based on knowledge about OS behavior, and some of them use combined features as well.

What's more, NAT devices are being widely deployed in the Internet nowadays. NAT detection is often considered important by ISPs since the share of one IP address among several devices may allow the unauthorized use of the network. The presence of NAT devices also breaks the end-to-end connectivity and make the network topology more complex.

There're several approaches about NAT detection problem. If more than one are devices detected from the same IP at the same time, it shows a great possibility of presence of NAT or tethering behavior. Many active or passive approaches were proposed to detect the presence of NAT or tethering. Tools [12, 1] we mentioned before are also able to detect NAT devices. TTL value in [11] or IPID sequences pattern in [4] was used as feature to count hosts and detect NAT devices. Some methods combined several features to improve accuracy. For example, Beverly in [5] developed a naive Bayesian classifier and used TTL, don't fragment (DF) and other field of TCP/IP header as features.

Most previous methods [12, 9, 2, 6, 11, 4] were based on traffic analysis. Especially they focused on HTTP traffic, which often incurs in-depth analysis on raw packets. However, raw packet capture is often deemed hard for a large-scale network.

In this paper, we resort to a novel method to fingerprint OS types by DNS logs. DNS service can be provided by network administrators or third-party providers. Any provider would have DNS requests logged when users use its service. Moreover, DNS [10] runs on application layer and need no changes of a running network. The logs are also easier to store than raw packets as well.

In our algorithm, we can successfully fingerprint OS types by only using the DNS logs. The previous work based on traffic analysis focused on "how" a device transmit a packet, and in our paper we're interested in "what" do they transmit. Since the algorithm using DNS log is behavior based, methods that aim to defeat or frustrate the OS fingerprint algorithm by manipulating the packets' format will not succeed anymore. And furthermore, we'll study about NAT based on the method and result of OS fingerprinting.

2. OVERVIEW OF PROPOSED METHOD

The basic idea to fingerprint the OSes using DNS log is, typically, most modern operating systems may use os-specific DNS queries to implement tasks like achieving the latest OS patch. Also, some operating systemes may have some os-specific applications to incur os-specific DNS queries, for example, iMessage in iOS systems. Thus we can infer the OS types by this knowledge.

To extract these OS specific features, we borrow the text categorization concepts.

First of all, we use the domain name field of the DNS log entry as our training sample, since the domain names may carry information about the OS it comes from. It represents the activity of the client. In this context, a user's activity can be viewed as an article, while each queried domain name can be regarded as a word in such an article. We try to extract feature from all those articles and select the best ones by using some feature selection algorithm of text categorization in section 3.1. Then the OS fingerprinting problem is equal to classify a new article of these feature domain names. We build a simple classifier in section 3.2 to deal with this problem. It is demonstrated that by only using a small group of domain names we can fingerprint OSes easily and accurately. As for NAT, we consider it a multi-class classification problem. We'll discuss it in section 4.2. At last, the paper is concluded in section 5.

3. ALGORITHM

3.1 Feature Extraction

Devices running different OSes may have different access behavior. Based on this observation, we train DNS log labelled by DHCP fingerprinting and extract the features for each category of OS.

For a specific operating system, it will resolve some specific domain names to implement some necessary tasks such as fetching latest update patch or clock synchronization. Some os-specific applications with auto update or sync function may also generate particular DNS queries. Moreover, users using different OSes may have different preference. The task is thus to find the OS specific DNS queries.

As we discussed in section 2, we regard a device's DNS log as an articile. Then we choose Chi-squared test as our feature selection methods. Since it has a clear mathematical meaning, is easy to implement, and of low complexity and good effect.

For a domain name and two OS categories, the contingency table is shown in table 1. a, b, c, d are number of IP addresses. And the score is calculated using formula (1). Then for a domain name d and an OS category os_0, we further use value minimum value of C' of os_0 and each other OS category (shown in formula(2)). It represents the "worst possible" discriminatory power of d for os_0.

At last we score and sort every domain name in the DNS log for each OS automatically by C_{d,os_0}. In this way we can find "the most representative domain names for each kind of operating system". We use these domain names as our features. We call the set of features representing an OS category os_0 as "a feature set" of os_0. We also make sure that each feature set has the same size.

$$C'_{d,os_1,os_2} = \frac{(ad-bc)^2(a+b+c+d)}{(a+b)(c+d)(a+c)(b+d)} \tag{1}$$

$$C_{d,os_0} = \min_{os_i \neq os_0, os_i \in C} C'_{d,os_0,os_i} \tag{2}$$

	has domain d in its DNS log?	
	yes	no
belongs to os_1	a	c
belongs to os_2	b	d

Table 1: Contingency Table

3.2 Classification

There have been a lot of classifiers designed for various purposes. Naive bayes, decision tree or k-nearest neighbors algorithm are all capable of fingerprinting the OS after the features have been selected.

Since the whether a domain name appearance in one IP address's DNS log is a simplest yes-no question, hamming distance may be appropriate for our classifier.

We define the feature set of OS_i as S_{OS_i}, a remote device needed fingerprinting as dev_j, and the set consists of dev_j's domain records as P_{dev_j}. We compute the intersection of set S_{OS_i} for each OS_i, then we choose the OS with the biggest cardinality of the intersection. In other words, for a device dev_j, we identify dev_j's OS as OS_0 when it meets the following equation.

$$\begin{aligned} OS_0 = & arg \max_{OS_i} |S_{OS_i} \cap P_{dev_j}|, \\ & OS_i \in \{android, linux, mac, win\} \end{aligned} \tag{3}$$

4. EVALUATION AND DISCUSSION

4.1 OS Fingerprinting

Though the concept of the classifier is simple, the result in figure 1 clearly shows that our approach is functioning in OS fingerprinting problem.

We use DNS log dumped from the DNS resolver of Tsinghua University. Many of devices in Tsinghua University use this server as their default DNS server, and

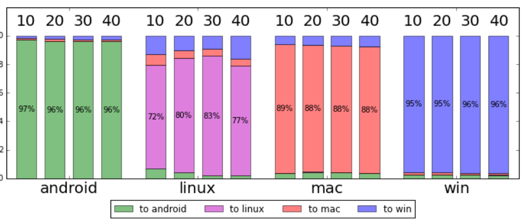


Figure 1: Evaluation in Different Amount of Features

the campus DHCP service will provide the address of it as the default DNS resolver.

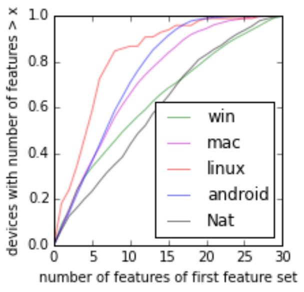
We run our algorithm on this data set. It has more than 0.95 recall rate for Windows and Android, 0.88 for Mac/iOS and 0.83 for Linux when feature number is 30. More detailed analysis indicates that the unbalanced data has an bad effect on the Linux's accuracy.

The number above the bars are the number of feature of each OS we use in our classification process. The result hardly changes when the cardinality of feature set changes, shows that our approach has high tolerance for number of features selected.

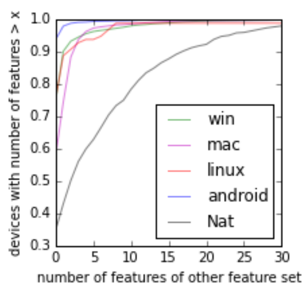
Above all, the features we've selected before have been proved effective in OS fingerprinting. And in this way we can identify an OS on by the presence of about 30 domain names.

4.2 Discussion on NAT or Tethering

Furthermore, as we mentioned in section 1, DNS data can also be used in tethering/NAT detection.



(a) First feature set



(b) Other feature sets

Figure 2: CDF of Number of Feature Domain Names in Device's DNS Log

Figure 2 shows some statistics about feature domain names. More specifically, it's the CDF of the number of feature domain names included in a device's DNS log. "First" feature set represents the feature set which has the biggest intersection with device's domain log(or the smallest hamming distance). "Other" feature in subplot 2b shows number of all other 3 kind of features. "NAT" curve represents the DNS log queried by the devices which are identified by DHCP fingerprinting as home routers of TP-LINK, NET-GEAR or other brands. Other OS are also identified by DHCP method, which means that it's the operating system running at device on network access point.

Differences between NAT and non-NAT devices can be observed clearly in figure 2b. According to the figure, about 90% of devices running a normal OS has less than 2 features from other feature sets apart from the first one, which is less than 50% for NAT devices. The results showed statistic significant difference, that is, for a normal operation system, it has a great probability that it only contains domain names from only one feature set of an OS, as the curve increase steeply at very beginning. And NAT devices are more likely contains several kinds of feature domain names than a normal non-NAT device. This knowledge can be used in NAT classification.

We use the similar way to the approach in OS Fingerprinting, except it is kind of multi-class classification problem. The main conception here is Tethering/NAT behavior often indicates the activity of sharing a connection. In this way, different OSes behind one single IP address reveals the presence of NAT/tethering behavior.

Therefore, we will check whether there are some domain names from the same IP address are likely generated by two or more different OSes within a short period of time. If that happens, we consider it a NAT device.

In our data set, there're 1466 of 19912 samples are classified as NAT by our algorithm. And about 730 of them are labelled as Windows by DHCP fingerprints, which is interesting. Some DNS queries specific to mobile OSes like Android are sent from them, which implies that laptops may be used as a Wi-Fi hotspot for mobile devices. This can hardly be observed in Android category(about 1%), which reveals that although most mobile devices has tethering function that allows them to share the Internet connection with laptops, people prefer to use notebook devices as hotspot.

Only 311 of them are labelled as NAT devices of popular brands such as NETGEAR or TP-LINK. And obviously, compared to general cases outside the campus, the NAT coverage is much lower in the university because IP address is enough that almost all devices can own a global one.

There're still a amount of clients that cannot be ignored choose to use NAT, anyway. It indicates that besides saving one or two global IP addresses, sharing connections with other devices conveniently and quickly also seems attractive to the end-users. People may use laptop to share Internet connection with their mobile devices in absence of fast mobile connection. That indicates the necessity of mobile devices and services nowadays as well.

4.3 Comparison Between DNS And DHCP Approach

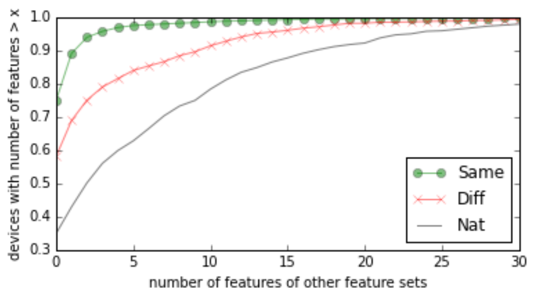


Figure 3: CDF of Number of Features When DNS & DHCP Approaches Differ.

In section 4.2, we've talked that DHCP can infer the OS of device at the access point. On other hand, our method based on DNS demands queries from all possible devices using that specific IP address. Therefore, there may be some distinction since the measurement objects of two methods are slightly different.

Figure 3 shows the CDF of number of features of "other" feature set when our approach based on DNS and labelling algorithm based on DHCP stay the same or differ from each other. The green curve on the top is when DNS and DHCP make the same decision. The red curve on the middle is when they differ. The black one on the bottom is the curve about Nat devices. It reveals that when they have different opinion about a device's OS, it is more likely that the DNS queries from that IP address are from more than one operating system. It shows the limitation of DHCP fingerprint. It is imprecise sometimes because the activity that PC or laptop running a Windows/Linux share Internet connection with other devices such as mobile phones can't be detected. That influences the accuracy of OS fingerprinting and NAT detection, and the accuracy of our approach, too. Since we assume that the DHCP fingerprint is accurate to label our data set.

5. CONCLUSION AND FUTURE WORK

In this paper we proposed a new OS fingerprinting method by domain names in DNS log. We extract feature domains from labelled DNS log for each OS. Then we build a classifier to identify the OS with these features before. We evaluate the result with data set from a large network. It can detect Windows/Android with the recall more than 0.95, 0.89 for Mac and 0.85 for Linux.

As for NAT, more analysis reveal that currently windows systems are more often used in NAT/tethering, which reveals end-users' preferences to use their "heavyweight" devices to share connection with their mobile ones for easy and fast Internet access. It indicates the ever-lasting life of NAT devices as long as these demands exist.

Preliminary results have demonstrate the algorithms effectiveness. However, there are still more researches required. For example, though DHCP logs are used to build the labelled set, it may be not precise sometimes. More information in DNS log such as port number or record type is not considered yet. They may also be helpful for OS fingerprinting or NAT detection.

6. REFERENCES

[1] Nmap - free security scanner for network exploration & security audits. http://nmap.org/.

[2] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel. Fpdetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1129–1140. ACM, 2013.

[3] O. Arkin. A remote active os fingerprinting tool using icmp. *login: the Magazine of USENIX and Sage*, 27(2):14–19, 2002.

[4] S. M. Bellovin. A technique for counting natted hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 267–272. ACM, 2002.

[5] R. Beverly. A robust classifier for passive tcp/ip fingerprinting. In *Passive and Active Network Measurement*, pages 158–167. Springer, 2004.

[6] Y.-C. Chen, Y. Liao, M. Baldi, S.-J. Lee, and L. Qiu. Os fingerprinting and tethering detection in mobile networks. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 173–180. ACM, 2014.

[7] L. G. Greenwald and T. J. Thomas. Toward undetected operating system fingerprinting. *WOOT*, 7:1–10, 2007.

[8] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. *Dependable and Secure Computing, IEEE Transactions on*, 2(2):93–108, 2005.

[9] J. P. S. Medeiros, A. M. Brito, and P. Motta Pires. A new method for recognizing operating systems of automation devices. In *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pages 1–4. IEEE, 2009.

[10] P. Mockapetris. Rfc 1035: Domain names - implementation and specification, november 1987. *URL http://www. ietf. org/rfc/rfc1035. txt*, 1987.

[11] K. Straka and G. Manes. Passive detection of nat routers and client counting. In *Advances in Digital Forensics II*, pages 239–246. Springer, 2006.

[12] M. Zalewski. p0f: Passive os fingerprinting tool, 2006.