A Hybrid Packet Clustering Approach for NAT Host Analysis

[1]Bo Zhang, [2]Yangyang Guan, Wenjia Niu, Jianlong Tan
Institute of Information Engineering
Chinese Academy of Sciences
Beijing 100093, P.R. China
email:[1]zhangbo@iie.ac.cn, [2]guanyangyang@iie.ac.cn

Zhi Mao
College of Information Engineering
XiangTan University
Xiangtan 411100, P.R.China
e-mail:172287369@qq.com

Abstract—**For a number of reasons, including the shortage of IPv4 addresses, many hosts are connected to the Internet through NAT devices. NAT devices effectively anonymize the origin of communication traffic, and remove many identifying features, which makes it difficult to isolate web traffic into mutually disjoint same-host sets called packet groups. However, ISPs or network supervisors can do product suggestions, targeted advertising, and online criminal detection by analyzing the packet group of one host. In this paper, we provide a hybrid packet clustering approach that can cluster NAT host's packets into packet groups. Our hybrid approach could group an NAT host's traffic from visiting different websites and would not suffer from bad network communications or anonymizing behavior of NAT devices. We use our isolating method on datasets obtained from two local area networks and both of them can get good results that accuracy is more than 90% and coverage is more than 50%.**

Keywords-NAT; packet clustering; third-party cookie ID; HTTP requests

I. INTRODUCTION

It is in the interest of ISPs and network supervisors to track the usage patterns of client hosts. This tracking allows them to understand user behavior for supporting applications such as product suggestions, targeted advertising, and online criminal detection. What the trouble is that private networks created by NAT devices make the IP address a public access point shared by tens of hosts. NAT devices are widely used by families, companies, hotels with the explosive increment of portable terminals like smart mobile phones, laptops.

Our work is to isolate http traffic standing on the position of ISPs and network supervisors. ISPs or network supervisors can get all web traffic from the internet which contains more information than what web services could obtain. Given a raw trace of web traffic collected from the outside of a private network, traffic isolating can be expected to take three steps:

1. Reconstruct TCP/IP connections from raw packets
2. Cluster connections into groups (mutually disjoint same-host sets)
3. Analyze each group to gather intelligence

There are some passive methods could be used to isolate NAT host's traffic, like IPID pattern, which exploit the fact that on many operating systems, the IP header's ID field is a simple counter[1], link chaining which links http packets by tracing hyperlinks[2].

The IPID method is unreliable for users' IPID sequence can be easily affected by bad network communications, communications with hosts behind the same NAT devices. What's more, only hosts running on windows systems regard IPID as an IP packets counter. All of these make the IPID method invalid. As for link chaining, it is hard to link HTTP packets that are generated by different websites, because visiting two different sites are usually not connected by hyperlinks.

In this paper, we provide a hybrid approach to isolate internet traffic of NAT hosts by two times of clustering: one-click clustering and cookie ID clustering. Our approach would not suffer from bad network communications, anonymizing behavior of NAT device and could connect packets generated by visiting different websites. Firstly, we analyze the behavior of hosts and build a modal for hosts surfing on the internet. In this modal, the time interval of visiting one web page is tiny which we called a time window. In each time window, we could use some rules to group packets from clicking one web page by a host which is called one-click clustering. Secondly, in cookie ID clustering, small packet groups are merged into bigger ones through many unchangeable cookie IDs. We find that like first-party cookie id, some popular third-party cookie ids like CNA, BAIDUID are also unchangeable whenever they appear. After the two steps of http clustering, we can divide all the http packets sent by a NAT device into tens of packet groups. Thirdly, we build two datasets from two local NAT networks. Based on our datasets which are labeled by intranet source IP, we get an average accuracy more than 90% for all the packet groups and an average coverage for NAT hosts more than 50%. Though our datasets are not large enough, the result is still inspiring.

II. RELATED WORK

Many efforts on tracking hosts focus on identifying specific hardware or packet characteristics, such as radio frequency [3], [4], [5], driver [6], timestamp [7]. However, this kind of identifiers are only contained in a few IP or HTTP packets, so that we should use more general identifiers that are much more widely observed in network traffic.

978-1-4799-1984-0 /15/$31.00 ©2015 IEEE

A widely observed host identifier is from browser. Peter Eckersley and his colleagues[8] investigate the degree to which modern web browsers are subject to "device fingerprinting" via the version and configuration information that they will transmit to websites upon request. They observe that the distribution of their fingerprints contains at least 18.1 bits of entropy, meaning that if we pick a browser at random, at best we expect that only one in 286,777 other browsers will share its fingerprint. As most of the http packages have a user-agent string, user-agent strings could be used to divide NAT traffic into groups each belonging to one host. However, the question is except user-agent other information like plugins, fonts, time-zone used by the fingerprinting by Peter Eckersley are not contained in most http packets captured external to NAT networks. What's more, although a fingerprinting only user-agent included still has a entropy 10.0, NAT users in the same private network like a company, cyber bar often share one or two kind of popular browsers like IE, chrome, safari.

Another widely observed host identifier is IPID [1], [9]. As shown in Fig. 1, each IPID sequence instance for one NAT host. However, there are some shortcomings about IPID sequence. Firstly, Communicating with servers or hosts from its private network, not being active continuously, IPID collisions can break up traffic from one host into many parts. Secondly, not all the operating systems share the characteristics described in [1], [9]. Lastly, some NAT devices have altered the IPID filed of IP header to a constant, which makes this technique useless.

Hyperlinks is also a very good element to connect http traffic. Constantine Daicos and his colleagues [2] show that passive external surveillance of web browsing hosts in private networks is possible despite the anonymizing effects of NATs and HTTP proxies at the gateway. Constantine and Scott offer us a content analysis technique called Link Chaining which is based on flowing hyperlinks in the bodies of HTTP messages; the technique is used to recover large pieces of user sessions called session fragments. Based on a large dataset, the experiment achieves a result that for a session fragment in average, the size is 10.62, the coverage is 12.63, the accuracy is 88.41. For one thing, isolating session fragments by analyzing hyperlinks in each http packet is time consuming. In our work, we only analyze http packets sent from NAT devices for traffic sent to NAT devices is huge. For another, our purpose also contains connecting sessions generated by visiting different websites independently for a NAT host.

Other identifiers [10], [11] like mail account, cookie ID are also popular in host tracking. Using application id like mail account, we can only isolate traffic of the application these ids (one application could contain more than one id, such as the application mail could have both mail account and cookie ID) belong to.

III. Problem Formulation

Our ultimate target is to establish a big data platform based on the internet that is used for host behavior analysis for ISPs and network supervisors. We think NAT devices are used everywhere with the development of Internet, so it leads to identify hosts by IP address invalid. One IP address often instances for more than one host in one house, hotel, office, etc. In our platform, we regard each host as a bunch of ids like cookie id, application id, user-agent

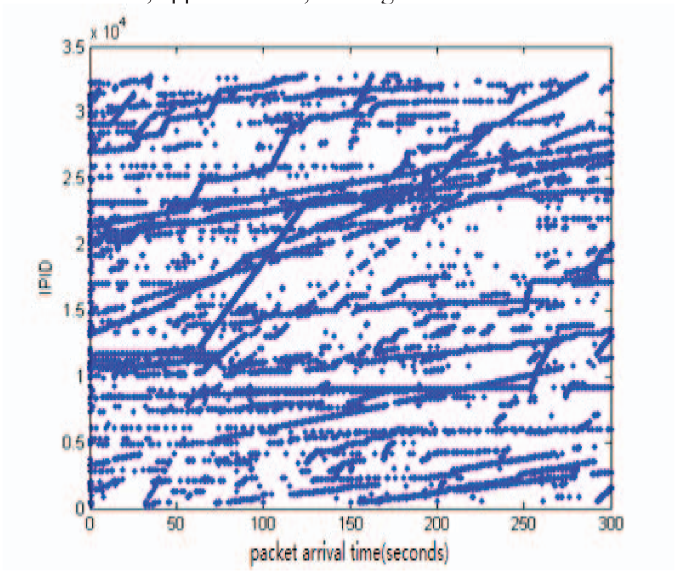

Figure 1. The IPID sequences of hosts in one of our NAT networks.

Other identifiers [10], [11] like mail account, cookie ID are also popular in host tracking. Using application id like mail account, we can only isolate traffic of the application these ids (one application could contain more than one id, such as the application mail could have both mail account and cookie ID) belong to.

Our approach will be used for user recommendation by ISPs. One ISP uses it to issue advertisements on its website and some associated web services. Firstly, we input all the HTTP data up till now to our approach thus getting many packet groups. Secondly, we extract ids (e.g., cookie id, application id) from each packet group, so we know some ids belong to the same host. Thirdly, we dig out what a host searches, buys, plays, etc. from each packet group. Lastly, when a host visits a website carrying some IDs we already know, we will recommend it some advertisements. Our approach could also help network supervisors to track hosts that may be criminals. Using the ids extracted from a packet group, network supervisors could know many activities of one user.

In this section, we will talk about some features of the internet traffic.

A. *Modeling a host visiting websites*

When a host surfs the internet, its behavior can be depicted as Fig. 2. Each rectangle means clicking one web page of a website. In this example, host h1 clicks a web page of website A during time range [t1, t2]. After a while, h1 clicks another web page of site A during time range [t4, t5]. The time gap between clicking two web pages of the same or different website is a few seconds to hundreds of seconds. Here, A and B are different websites means that neither site B is the third-party site to A nor A is to B. Both

A and B are first-party websites with topic like news, games, Search Engines, etc. Usually, some of the http requests produced by clicking a web page are sent to some invisible websites which are called third-party websites by us in this paper.

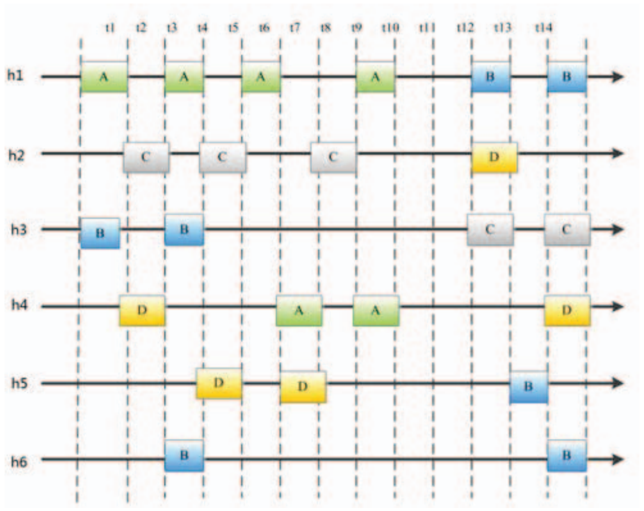

Figure 2. The behavior when host surfs on the internet. Each rectangle instance for a clicking on one web page.

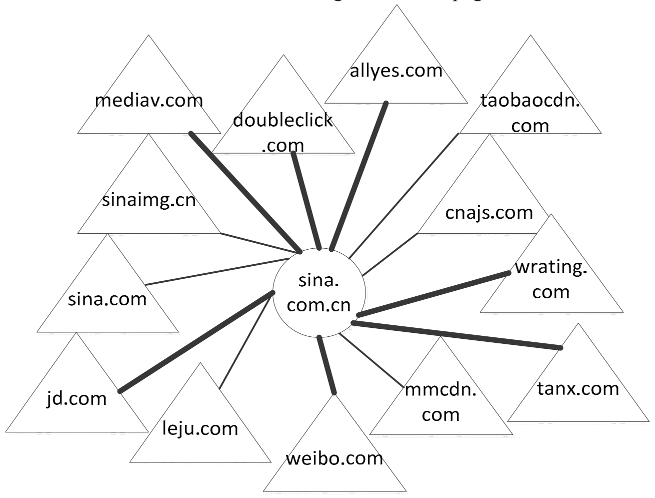

Figure 3. Many http requests are sent to third-party websites when clicking a web page of a first-party website.

TABLE I. SOME POPULAR THIRD-PARTY WEBSITES AND THE CORRESPONDING COOKIE IDS.

1	cnzz.com	cna
2	admaster.com.cn	BAIDUID
3	miaozhen.com	mz_id

B. *Third-party websites are popular*

Fig. 3 is generated by Lightbeam, a tool made by firefox [12]. The circles identify a visible website (first-party website) you visited, the triangles an invisible website (third-party website). When you visit a website, some third-partywebsites are also visited at the same time. The lightbeam graph was generated when homepage was visited and the visit was nearly within a few seconds.

Invisible websites have been popular from at least several years ago. It depicts that when visits 1075 alexa websites, more than 50% are associated via top 10 invisible websites [13]. The time of this statistics is 2005 -2006 which means invisible websites is more popular now. It is shocking because it means that some dominant invisible nodes may track you with cookies when you search, shop, read news, play games on the internet. Maybe in the database of one invisible website, there is a profile about all your things, like
hobbies, age, education background, families, sex orientation. The invisible node is the one who know you best instead of you.

C. *Fixed third-party cookie IDs*

HTTP Cookies, also known as Web cookies or just cookies, are small parcels of text sent by a server to a web browser and then sent back unchanged by the browser if it accesses that server again [14]. Cookie IDs are those that used for identify or track hosts. There are two kinds of Cookie IDs used by third-party websites from the position of gateways. One is fixed cookie ID. This kind of third-party Cookie IDs are persistent no matter what the first-party websites are. The other one is varied cookie ID. For this kind of cookie ids, only the third-party websites themselves know how to map their cookie IDs together. Table 1 displays some third-party websites and their cookie IDs. We extract them from the traffic generated by visiting some popular websites. There are some third-party cookie IDs like BAIDUID, CNA that each appears in more than one third-party domains. The reason may be that there is Commercial cooperation between websites share the same cookie ID.

IV. ALGORITHM

With the help of the features described in the last part, we can use some association rules to cluster http packets belong to the same host. First of all, we merge the http packets belong to one-click traffic to one group which we can called one-click group. Secondly, we merge one-click groups by first-party cookie IDs. After this step, http traffic belong to visiting the same first-party website could be associated together. Thirdly, third-party cookie ids would be applied to connect http traffic from visiting different first-party websites. With first-party cookie Ids, we can connect traffic generated by visiting one website at different time.

Though cookie ID is very reliable for associating, we could not apply cookie IDs directly in the isolating process for it is impossible to get a cookie ID table that contains all

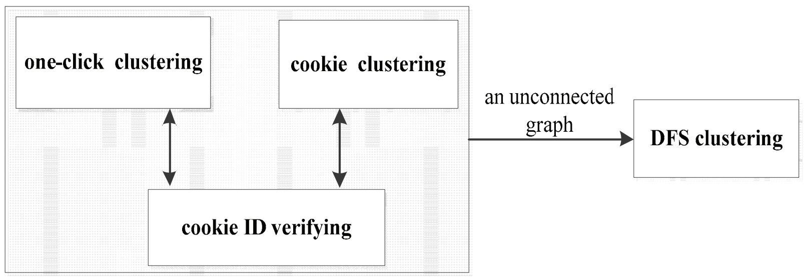

Figure 4. The isolating process of our approach. DFS(depth first search) algorithm is used to get each connected component of the graph.

the cookie ids in the internet. Therefore, we use a little trick here. Two https that contain a same cookie ID domain but different cookie ID values do not belong to one host. In addition, we remove the dependency relationship between one-click clustering and cookie clustering. Therefore, our isolating process can be depicted as Fig. 4.

In our approach, the outputs of one-click clustering and cookie clustering are edges and each edge indicates two http requests are from the same host. Using the edges, we could build an undirected graph which has many connected components. Each connected component indicates http packets sent by the same host. We use a adjacency list to store the undirected graph and execute the DFS (depth first search) algorithm to get the result.

A. One-click clustering

In order to group one-click http requests together, we use a technique called sliding time window. In each time window, combined with cookie ID verifying, we use some rules which can mostly guarantee two http requests belong to the same host. All the rules employed in each time window are observed by analyzing http headers. In this paper, the time sliding window size is used as a constant which is between 2000 to 4000 milliseconds.

1) |T(a)-T(b)|<Delta-T: The timestamps of http request a and b should be within the same time window of which the size is Delta-T. A web page always contains a few parts which would lead to more than one http requests being sent by a browser. By testing in some NAT networks, we find that almost all the http requests generated by one-click are sent out within 2000-4000 milliseconds. This is how we get the sliding window size.

2) UA(a) =UA(b): User-agent is unique for every browser and it is an important identification to track hosts [8], [10]. The Http requests generated by one-click have the same user agent string. Timestamps of http requests a and b should be within the same time window of which the size is *Delta-T*.

3) src_ip(a)=src_ip(b): Many NAT devices would use dynamic IP which are common in the internet. This rule means that during a time window, all the Http requests of one-click must have the same source ip. For traffic of a NAT host behind a dynamic NAT IP, the association will be completed by the cookie ID clustering part.

4) host(a)=host(b)||ref(a)=ref(b)|| dst_ip(a)=dst_ip(b) || ref(b) = host(a) =UA(b): The destination of a and b are the same means that their host in http headers or destination IP in IP headers are identical. If a and b are sent to the same host during a tiny time window, they are from the same host probably. References of a and b http headers are identical means that they are hyperlinks from the same page, which is also an important same-host proof. As for ref(b) = host(a), it indicates b is a hyperlink of a.

Though rules listed above could not be used to guarantee http requests a and b belonging to the same host, combined with cookie ID verifying, these rules together can achieve a good result.

B. Cookie clustering

In this part, we use cookies instead of cookie IDs to do the association. From our observation that though most of cookies are not cookie IDs, a large part of them would not change within a short time. These short-time unchangeable cookies may be session cookies which are also known as an in-memory cookie or transient cookie, exists only in temporary memory while the user navigates a website [14]. This kind of transient cookies can be applied to improve the coverage of our association. However, a fact is that some cookies which are not session cookies like city location, reference host would exist in the http requests of many http hosts. Other cookies are hardly understood for us, but they are always very short, variable frequently. Cookies that are not cookie IDs could not be used as credible identifications to connect two http requests because they are not hosts exclusive. However, when connect two http requests by cookies that are not hosts exclusive, cookie id verifying would be helpful. In this paper, We only keep those of which the cookie (cookie name + cookie value) are longer than 12 characters for too short cookie values are possible to applied by different websites.

C. Cookie ID verifying

When connect two http requests by one-click clustering or cookie clustering, cookie ID verifying would work to check if there is cookie ID confliction between them. First of all, we should create a cookie ID table which contains cookie ids of all the websites. In order to extract a website's cookie ID, we use the technique employed by Chuan [15]. For each website, we would visit it two times and the time interval between two visits is at least 24 hours. By analyzing http generated by visiting a website, it is easy to find out those unchangeable cookies. In addition, we only pick the most probable cookie of which most of characters are digit and alphabet as cookie ID.

Then how does cookie ID verifying work? When connect two http requests A and B, if requests A and B have a common cookie ID name but different cookie ID value, we say there is a cookie ID collision between A and B.

Why is cookie ID verifying essential? For one thing, in one-click clustering part, it is possible that host h1 and host h2 having the same user-agent visit the same web page within a time window. If there is a cookie ID confliction between two http requests having user-agent U in one time window, all the merging would be stopped among http requests having user-agent U. This can prevent incorrect merging traffic of h1 and h2. For another, in cookie clustering part, because it is impossible to create a cookie ID table covering all the websites in the internet, besides cookie ID, we use any other cookies to do the connecting. In one http, there is often more than one cookies and only one cookie is cookie ID. Cookie ID verifying prevents connecting two http requests with a common cookie (the cookie is not cookie id) but have a cookie ID conflicting. Using cookie ID as verifying rather than connecting by cookie ID, more times of merging are done but the accuracy of merging group is lower, because cookie ID verifying could not eliminate all the incorrect connections.

TABLE II. TIME COMPLEXITY AND SPACE COMPLEXITY

Modal of our approach	Time complexity	Space complexity
one-click clustering	n	Buffer size
cookie clustering	m*log(m)	Buffer size
DFS clustering	m	m+n

```
http requests in **File** is in order by timestamp
list before_cur = null;
while(read elem from **File**)
    cur = elem;
    add(cur) to before_cur
    for(it in before_cur)
        if(timestamp(cur)- timestamp(it) > DELTA-T)
            erase(it) from before_cur
        else
            if cur and it belong to one-click page
                output an edge <id(cur), id(it)>
```

Figure 5. one-click clustering algrithm

D. *Time and space complexity*

Fig. 5 illustrates one-click clustering algorithm. Using a buffer prefetching source data from log file, we only need to put a small part data in the memory, of which size is the buffer size. As for cookie clustering, we extract an element <cookie, cookie ID, id> from each http request. After sorting all the elements by the column 'cookie', the clustering process will be very easy. Table 2 depicted the time and space complexity of all the three clustering parts. 'm' is the number of edges and 'n' is the number of http requests. With the aid of data compression, our approach could be more fast and memory saving.

V. DATASETS AND VALIDATIONS

After Network traffic was collected passively from the inside of two live laboratory networks with a high volume of http requests traffic and later written to a Log file. The traffic capturing position is situated at the gateway before any NAT so that intranet IP addresses are visible. The real working position of our approach would run external to NAT devices, but IP address visibility is necessary here to validate the results. The two datasets we used here are illustrated in table 3. The data are filtered by removing http requests that do not contain cookies and user-agent. Http requests not having cookies are considered as valueless by us. As for http request not having user-agent, they are sent by some specific applications instead of browsers in the hosts and they are not appropriate for our approach now.

The isolating output of our approach is a bunch of http requests groups and we call them data fragments. For each fragment, we want to use it for extracting user habits and hobbies to creating host profile on the internet, so the fragment must be as large and pure as possible [2]. We use some measures to reveal the effect of our approach. We use the definitions of coverage and accuracy invented in [2]. In addition, we make average host accuracy and average fragment cover as overall indexes of our approach.

Fragment f is a packet group that generated by our isolating approach and s is the http traffic sent by the host that f belongs to. When we say f belongs to s, we mean that more than 50% of the http requests of f could be attributed to s. Those fragments that could not be attributed to any host would be discarded. Average fragment Accuracy is defined as the average accuracy of all the fragments and $A(\text{fragment}_K)$ is the accuracy of fragment_K. Average host coverage is defined as the average coverage of all the hosts and $C(\text{host}_K)$ is the coverage of host_K. $C(\text{host}_K)$ is the coverage of the fragment that has the biggest coverage among fragments belonging to host_K. The reason why we do not use average coverage of all the fragments is that if a host has two fragments generated by our approach and one has coverage of 90% and another 10%, average coverage of the two fragments would be 50% which is not what we want.

fragment Accuracy $$A = \frac{|f \cap s|}{|f|}. \quad (1)$$

fragment Coverage $$C = \frac{|f \cap s|}{|s|}. \quad (2)$$

N_f and N_h are regarded as the number of fragments and hosts in our experiment.

TABLE III. THE DETAILS OF THE TWO DATASETS WE USED

dataset	Total http requests	time	Active hosts	Visiting Sites/host	Kinds of UAs	UAs/host	Hosts/UA
NAT-1	106234	24 hours	83	86	417	10.0	2.1
NAT-2	396132	24 hours	118	176	580	7.0	1.95

TABLE IV. ACCURACY AND COVERAGE OF DATASET1. THIS TABLE ONLY SHOW THE GROUPS OF WHICH BOTH ACCURACY AND COVERAGE ARE NOT LESS THAN 0.5.

A | sum(groups) | C	1	1-0.9	0.9-0.8	0.8-0.7	0.7-0.6	0.6-0.5
1	5	6	7	5	5	14
1-0.9	0	0	0	1	0	1
0.9-0.8	1	0	0	0	0	0
0.8-0.7	0	0	0	1	0	0
0.7-0.6	0	0	0	0	0	0
0.6-0.5	0	1	0	0	0	0

TABLE V. ACCURACY AND COVERAGE OF DATASET2. THIS TABLE ONLY SHOW THE GROUPS OF WHICH BOTH ACCURACY AND COVERAGE ARE NOT LESS THAN 0.5.

A | sum(groups) | C	1	1-0.9	0.9-0.8	0.8-0.7	0.7-0.6	0.6-0.5
1	4	9	16	11	4	20
1-0.9	0	0	0	1	0	1
0.9-0.8	0	0	0	0	0	0
0.8-0.7	0	1	0	0	0	0
0.7-0.6	0	0	2	0	0	0
0.6-0.5	1	1	0	0	0	0

Average fragment Accuracy $AA = \frac{1}{N_f} \sum_{\kappa=1}^{N_f} A(\text{fragment}_\kappa)$. (3)

Average host coverage $AC = \frac{1}{N_h} \sum_{\kappa=1}^{N_h} C(\text{host}_\kappa)$. (4)

In our experiment, many groups generated by our isolating process are very small. These trivial groups are more than 50% of the total groups. In the validation part, we discard the groups that only have a single http requests. Though the number of trivial groups is large, the total amount of http requests they contain is very small.

Table 4 and table 5 are some results of the two datasets executed on our isolating process. As we can see, there are 47 groups totally in table 4 and both the accuracy and coverage of each group is more than 50%. The total number of dataset2 is 73.

Moreover, the average fragment accuracy of dataset1 and dataset2 is 0.994 and 0.982, while the average host coverage of them is 0.5437 and 0.587. Our approach can isolate hosts' http traffic well standing on the outside of a NAT device. From table 3, we can see during the 24 hours, while each host visits more than 80 sites (first-party and third-party sites) averagely, the AC should be more than 50%. All these can be attributed to the rules used in each sliding time window and http cookies.

Dynamic IP addresses are used by many hosts, among which some are portable devices like smart mobilephones and others are Nat hosts behind NAT devices holding IP addresses configured dynamically by ISPs. In our experiment, one-click clustering uses a time window which is so small that two NAT hosts using a common browser in the same NAT network could not be clustered as one. Cookie clustering can group traffic sent by a host behind a NAT device using variable IP address, because cookies are irrelevant with IP address. Therefore, on the gateway of ISPs, our approach could handle the situation that dynamic IP addresses with hosts are popular.

We can extract a bunch of cookie ids from each clustering group generated by our approach. From these ids, we can learn what the host search, what movies or news it watch, even what goods it buy. Some search engines like baidu, 360, google use simple bytes conversion to transmit the keywords users search, so it is easy to learn what users search by analyzing internet traffic. URLs contain topics of news or movies usually which are easy to extract by some simple tools. As for what users like to buy in e-commerce websites, the URLs also contain the names of shops or commodities.

However, there are still some limitations of our approach:

1) cookie churning

Some users like to clear cookies in their hosts periodically which could bring some troubles to our isolating process. Removing cookies is regarded as one kind of cookie churning [10] and it can totally decrease the coverage of our isolating result.

2) multi-browsers hosts

One-click clustering and cookie clustering could not connect requests sent by two different browsers belong to one host. In our datasets, hosts having more than one browser are very common.

3) our datasets are small

For it is impossible to construct a dataset consists of millions of hosts and sites, our validations may still be experimental.

VI. CONCLUSION AND FUTURE WORK

In this paper, we demonstrate a approach that is used to isolate NAT hosts' traffic. The approach can be used by ISPs or network supervisors to understand user behavior for supporting applications such as product suggestions, targeted advertising, and online criminal detection. The main techniques we used here are sliding time window and cookie clustering. Each of our two datasets is from NAT network which consists of about 100 hosts; the accuracy is more than %90 and the coverage is more than 50%. Third-party cookies are the bridges that connect traffic generated by visiting different first-party websites.

Some information in packet groups used in [8] could be applied to connect groups that sent by the same host but separated by cookie removing. As for multi-browsers hosts, People may login their webmail or some websites by different browsers. Therefore, one method to group two browsers' traffic together is using user id as connections. At last, we will construct a labeled larger dataset from http requests grabbed in the gateway of ISPs. Then we extract those requests belong to IPs that only have one user-agent and regard the filtered IPs as hosts behind a common NAT device.

REFERENCES

[1] S. M. Bellovin, “A technique for counting NATted hosts,” Proc. of the 2nd ACM SIGCOMM Workshop on Internet measurement, Nov. 2002, pp. 267-272

[2] C. Daicos, and S. Knight, “A passive external web surveillance technique for private networks,” In Computer Network Security, Springer Berlin Heidelberg. 2005, pp. 88-103.

[3] J. Hall, M. Barbeau, and E. Kranakis, “Detection of transient in radio frequency fingerprinting using signal phase,” Wireless and Optical Communications, 2003, pp. 13-18.

[4] K. Rasmussen and S. Capkun. "Implications of radio fingerprinting on the security of sensor networks." Security and Privacy in Communications Networks and the Workshops, IEEE, 2007, pp. 331-340.

[5] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, “Wireless device identification with radiometric signatures,” Proc. of the 14th ACM international conference on Mobile computing and networking, Sep. 2008, pp. 116-127.

[6] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J.V. Randwyk, and D. Sicker, “Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting,” In Usenix Security, July. 2006.

[7] T. Kohno, A. Broido, and K. Claffy. Remote physical device fingerprinting. In IEEE Symp. Security and Privacy, 2005, pp. 93-108.

[8] P. Eckersley, “How unique is your web browser?” In Privacy Enhancing Technologies, Jan. 2010, pp. 1-18, Springer Berlin Heidelberg.

[9] S. Mongkolluksamee, K. Fukuda, and P. Pongpaibool, “Counting NATted hosts by observing TCP/IP field behaviors,” Proc. 2012 IEEE International Conference on Communications June. 2012, pp. 1265-1270.

[10] Y. Xie, F. Yu, and M. Abadi, “De-anonymizing the internet using unreliable IDs,” In ACM SIGCOMM Computer Communication Review, Vol. 39, No. 4, Aug. 2009, pp. 75-86..

[11] T. F. Yen, Y. Xie, F. Yu, R. P. Yu, M. Abadi, “Host Fingerprinting and Tracking on the Web: Privacy and Security Implications,” In NDSS, Feb. 2012.

[12] Lightbeam for firefox. <https://www.mozilla.org/zh-CN/lightbeam/>.

[13] B. Krishnamurthy, and C. E. Wills, “Generating a privacy footprint on the Internet,” Proc. of the 6th ACM SIGCOMM conference on Internet measurement, Oct. 2006, pp. 65-70.

[14] HTTP Cookie, <http://en.wikipedia.org/wiki/HTTP_cookie>.

[15] C. Yue, M. Xie, and H. Wang, “An automatic HTTP cookie management system,” Computer Networks, vol. 54(13), 2010, pp. 2182-2198.