一种基于 Cookie 的网盘资源在线溯源方法

罗攀[1,3,4]，袁庆升[2]，张鹏[3,4]，李舒[3,4]，郑超[3,4]

(1. 西安电子科技大学，西安 710071； 2. 国家计算机网络应急技术处理协调中心，北京 100029； 3. 中国科学院 信息工程研究所，北京 100876；

4. 信息内容安全技术国家工程实验室，北京 100093)

摘 要：网盘作为一种流行的资源传播方式，其所分享的资源已经在网络流量中占有越来越多的比例，因此获取网盘资源的分享链接对于网络安全有着重要的意义。为此本文提出了一种基于 Cookie 的分享链接获取方法—CookieTracking。该方法首先建立 cookie 和 HTTP 会话的索引。其次，通过 location 哈希表和 token 哈希表获取了下载网盘资源的 URL 跳转链。最后，通过 URL 跳转链上的每个节点的统计分析获取其中的资源分享链接。实验结果显示在带宽为 1Tbps 的网关上 CookieTracking 的平均查准率、查全率、获取时间分别为是 98.64%,98.35%,0.102ms，表明了该方法具有很好的性能和可扩展性

关键词：网盘资源；分享链接；URL 跳转链；cookie；HTTP 会话

A on-line URL tracking method based on cookie for Cyberlockers resource

LUO Pan[1,3,4], YUAN Qing Sheng[2], ZHANG Peng[3,4], LI Shu[3,4], ZHENG Chao[2,3]

(1.Xidian University, Xi'an 710071, China;

2. National computer network emergency coordination center, Beijing 100029, China;

3.Insititute of Information Engineering, Chinese Academy of Sciences, Beijing 100876, China;

4. National Engineering Laboratory for Information Security Technology, Beijing 100093, China)

Abstract: Cyberlockers have recently become a very popular means of distributing content, and increased usage of these services could drastically alter the characteristic of Web traffic. In light of the non-negligible fraction accounted by the traffic flows originating from Cyberlocks, it is necessary to track them for network security. In this paper ,we proposed CookieTracking ,a method to detect the shared links to the resource of cyberlockers. Firstly, we establish the indexing for the cookie and HTTP session. Secondly, we track the URL chain of downloading Cyberlockers resouse. At last ,we obtain the commen node for multiple URL chain of one downloading Cyberlockers resouse.we think this commen node is the shared links we wanted. The experimental results demonstrate the effectiveness and extensibility of our method ,at the gateway that bandwidth is 1Tbps , where the precision ratio, recall ration, time respectively has been reached 98.64%,98.35%,0.102ms.

Key words: resource of cyberlockers; shared links; URL chain; cookie; HTTP session

1 引言

网络作为一个开放式的平台，互联网上有一系列的让用户可以分享资源给他人的服务，例如过去比较流行的 p2p、Bittorrent 以及目前比较流行的网盘。由于网盘操作简单（一键分享下载以及用户无

基金项目：国家 242 信息安全支撑计划课题（2014A099）；

Foundation Items: National 242 Information Security support plan(2014A099) ;

需安装下载软件）和下载速度快（与 Bittorrent 等传统资源分享模式相比）的特点使 p2p 和 Bittorrent 使用量急剧下降[1.2.3]。很多学者对网盘的使用程度作出统计。Maier[1]统计显示，在普通网络流量中网盘流量占总流量的 17%。Gehlen[2]统计显示，网盘是排名前十的网络应用以及占据 5%的点击量。Allot[3]统计显示在移动终端上网盘流量占据总流量的 19%。当用户将资源上传至网盘并分享该资源时，网盘会给该资源生成唯一对应的 URL。用户将该链接分享至第三方网络社交平台（博客，论坛等），其他用户即可点击该链接下载分享资源，这些用户点击分享链接后会弹出一带有下载按钮的页面（该页面的 URL 即为分享链接），该页面会描述该下载资源的属性信息，例如资源发布者，资源发布时间，资源下载次数等等。当用户点击网盘资源页面的下载按钮下载该资源时，其浏览器会通过 URL 自动向服务器发出一系列 HTTP 请求，直至建立下载资源的 HTTP 会话。如何从骨干网络节点上海量的网络流量中获取网盘下载资源 HTTP 会话对应的网盘资源分享链接（本文将这一过程其定义为网盘资源溯源）对于网络审查[4]、网络取证[5]、网络流量监控[6]等具有重要意义。众所周知，referer 是 HTTP 表头的一个可选字段，用来表示 HTTP 来源地址，但是，在真实流量统计显示，只有 17.7%的 HTTP 会话存在 referer 字段。因此只依赖 referer 字段无法获取绝大多数下载资源 HTTP 会话的网盘资源链接。同时， NAT[7]、多路组播技术[8]、HTTP 代理[9]导致在公网路由节点捕获的 HTTP 会话的 IP 地址也不能作为精确追溯其 URL 跳转链的依据[10]。为此，本文提出了一种基于 Cookie 的网盘资源在线溯源方法—CookieTracking，具有以下三个创新点：

(1) 建立了 cookie 与 HTTP 会话以及 location 与 HTTP 会话的索引。
(2) 只需要缓存 HTTP 会话中的 cookie、URL 和 location 字段，大大降低了空间开销，能够适应于骨干网络节点。。
(3) 通过统计分析同一网盘资源对应的多条 URL 跳转链的唯一公共 URL 节点，获取网盘资源的分享链接。

2 相关工作

针对 URL 溯源，国内外学者的研究并不多。目前可用于网盘资源溯源的研究方法按可分为两类：

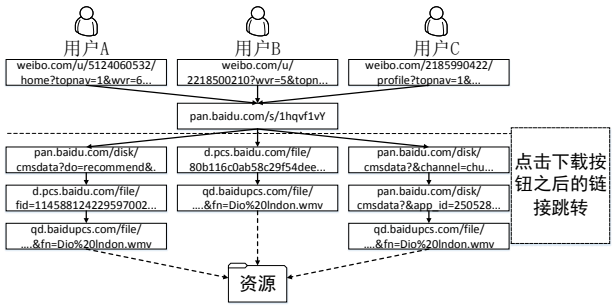
第一类是针对网页木马、恶意网页识别[11]而提出的 URL 跳转链入口 URL 识别方法。由于网页木马以及恶意网页等为了躲避爬虫软件的检测通常都会经过多次 URL 重定向将用户浏览器最终引导向恶意代码网页 [11]。这种 URL 多次跳转给网页木马的识别带来了困难，为此文献[11-17]提出了网页木马等入口 URL 识别技术。文献[11]针对 twitter 上存在的恶意 URL 识别提出 warningbird 方法。Warningbird 方法通过 twitter streaming APIs 收集带有 URL 的推文,找出同一分享 URL 的多条 URL 跳转链以及分享 URL 对应的入口 URL。通过特征提取、训练、分类获取了恶意 URL 的入口 URL 的特征。文献[12]针对网页木马识别提出了 Arrow 方法。Arrow 方法首先通过高交互式蜜罐收集同一恶意软件的不同 URL 跳转链。其次，对比 URL 跳转链节点的 IP 和域名获取恶意软件的入口 URL。最后，给入口 URL 生成正则表达式，根据正则表达式识别网页木马。文献[11-17]都是通过收集恶意代码网页的 URL 跳转链，离线学习入口 URL 的特征，根据这些特征实现恶意代码网页与入口 URL（恶意网页、挂马网页）的识别。这种方法同样可用于网盘资源的 URL 溯源，但是存在以下缺点：第一：目前的网盘种类多样导致不同网盘的 URL 跳转链特征并不一致，很难满足所有网盘的 URL 溯源。第二：通过调研发现即使对于同一网盘多次下载的特征也会变化。第三：由于 URL 特征的多变性，离线学习方法需要巨大的维护成本[15]。

第二类是针对 NAT 和 HTTP 代理导致骨干网关上数据包的 IP 地址无法标识用户而提出的在 NAT 主机识别技术[18-19]。其中，文献 [18]通过分析 html 网页内容，以及 HTTP 会话中的 user-agent 字段，实现了对不同用户发出的一系列 HTTP 请求的关联。文献 [19]通过对用户浏览器的版本和配置等信息产生“浏览器指纹”的方法，识别出不同用户浏览器所发出的的 HTTP 会话。这种方法的缺点是：第一：骨干网络大部分的 HTTP 会话中只包含 user-agent，而没有其他的配置信息（字体，插件，时间等）。第二：针对骨干网络的流量，缓存网页内容会极大的加剧空间开销。

3 CookieTracking 的原理

本文将用户点击分享链接，浏览器建立获取网

盘资源链接的 HTTP 会话，到建立下载资源 HTTP 会话为止，以及这期间浏览器发出的这一系列 HTTP 请求对应的 URL 定义为 URL 跳转链。对于同一分享链接，每一次下载资源对应的 URL 跳转链节点的数目，除网盘资源链接的 URL 相同外，其他 URL 节点都不相同。如图 1 所示案例是用户 A，B，C 点击网盘分享链接：pan.baidu.com/s/1hqvf1vY 下载资源所对应的 URL 跳转链。除分享链接外，其他的 URL 节点都不同。



基于以上特点，CookieTracking 的基本原理为：首先，从网关流量中识别下载资源并计算资源的标识 ID。然后，获取下载资源的 URL 跳转链。最后，合并具有相同资源标识 ID 的多条 URL 跳转链，获取唯一的公共 URL 节点，该节点即为该下载资源对应的分享链接。下面对这三部分做具体阐述：

3.1 识别下载资源并计算资源的标识 ID

3.1.1 识别下载资源

网盘资源传输的 HTTP 会话有以下特点：第一、下载资源 HTTP 会话的 content type 为：vedio/mp4、application/stream 等。第二、真实流量统计显示 93%的下载资源 HTTP 会话的 content length 都在 50M 以上 。因此可根据上述特点识别出所有包含网盘下载资源在内的下载资源。

3.1.2 计算资源的标识 ID

由于下载资源在网络上是按包传输的，在大流量环境中传统缓存整个下载资源数据计算 MD5 的方法一方面极大的消耗了内存资源，另一方面也增加了分享链接的获取时间。因此，并不适用于 CookieTracking。为此 CookieTracking 采用了累计哈希的方法计算下载资源的标识 ID。一方面，对于按包到达的数据只对包缓存，对每个字节累计进行哈希，最终将下载数据其映射成一个 64 字节的字符串。另一方面，真实流量中，下载资源的部分数据即可以对资源进行区分，因此，CookieTracking 方法只对下载资源的前 20%-30%数据做累计哈希。

3.2 获取下载资源的 URL 跳转链

下载资源的 URL 跳转链具有以下 3 个特点：（1）由于网盘分享链接允许借助第三方的下载工具下载，下载资源 HTTP 会话是 HTTP 重定向的结果，下载资源 HTTP 会话并无 cookie 信息。（2）网盘用户必须允许浏览器使用 cookie，因此除下载资源 HTTP 会话外，URL 跳转链的其他节点对应的 HTTP 会话都含有 cookie。此外，每个节点对应的 HTTP 话单的 Cookie 信息存在多个相同的 key=value 项（本文将其定义为 token）。其中某些 token 是网盘服务器用来追踪用户，于用户一一映射，可以标识用户的访问记录。为此本文定义了 token 的区分度：

$$dif = \frac{N_{token-cookie}}{N_{cookie}} \qquad (1)$$

其中 $N_{token\text{-}cookie}$ 为：包含该 token 的 HTTP 话单数。N_{cookie} 为总的 HTTP 话单数。

进一步，为了提高 URL 跳转链的准确性，cookieID 定义为两个 HTTP 话单的关联度：

$$sim_{token} = N_{simtoken} \qquad (2)$$

$N_{simtoken}$ 为两个 HTTP 话单的 cookie 值中高区分度 token 的个数。如 sim_{token} 大于某阈值 sim_0，即认为这两个 HTTP 话单属于同一条 URL 跳转链。

因此，只要获取与下载资源 HTTP 会话有着高关联度的一系列 HTTP 话单即可获取 URL 跳转链。

（3）根据 HTTP 重定向原理，下载资源 HTTP 会话的 URL 与重定向 HTTP 会话的 location 相同，而重定向 HTTP 会话存在 cookie 信息。因此，通过下载资源 HTTP 会话的 URL 获取重定向 HTTP 会话。然后，通过重定向 HTTP 会话即可获取完整的 URL 跳转链。

3.3 获取分享链接

将相同资源标识 ID 对应的多个 URL 跳转链合并，获取多个 URL 跳转链的唯一公共 URL 节点，该节点即为该资源的分享链接。

4 CookieTracking 的实现

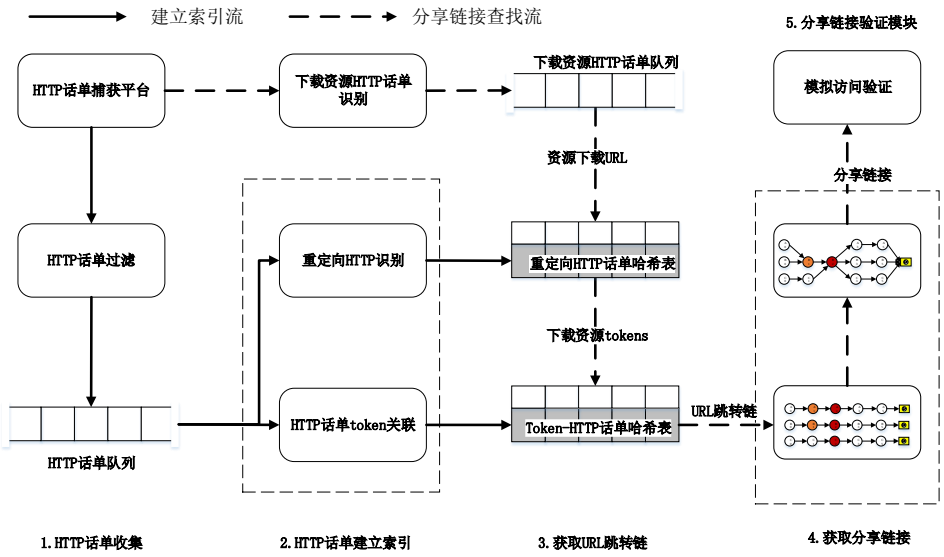


图 2　CookieTracking 实现架构图

如图 2 所示，为 CookieTracking 方法的实现架构。包含五个部分：1.HTTP 话单收集模块，负责对输入的网络流量通过流量捕获平台获取所需 HTTP 话单（为了降低空间开销，CookieTracking 只缓存 HTTP 会话的头部信息，将其定义为 HTTP 话单）；2.HTTP 话单索引模块，负责解析 HTTP 话单，对海量的 HTTP 话单建立 cookie 字段与 HTTP 话单的关联；3.获取下载资源 URL 跳转链模块，负责根据下载资源 HTTP 话单获取重定向 HTTP 话单，根据重定向 HTTP 话单 cookie 获取 URL 跳转链；4.分享链接查找模块，合并同一下载资源的多个 URL 跳转链，获取分享链接。 5.分享链接验证模块，负责比较分享链接下载资源的标识 ID 与 loadrunner[21]模拟访问收集的资源标识 ID，比较分享链接正确性。以上 5 个模块均并行执行。下面具体介绍每个模块的实现细节：

4.1　HTTP 话单收集模块

该模块通过网络流量处理平台解析 HTTP 流量。首先，过滤出需要的两类 HTTP 会话信息：

（1）如果 HTTP 话单的 content type 为 text/html，且存在 cookie 字段，将这类 HTTP 会话信息的三元组信息：（URL、cookie、TCP 连接建立时间戳）缓存于 HTTP 话单队列。

（2）如果 HTTP 话单的 content type 为 video/x-ms-wmv、video/mp4 等音视频 MIME 类型，且 该 HTTP 话单的 content length 大于某阈值，则该 HTTP 话单为资源下载 HTTP 会话。将这类 HTTP 会话的四元组信息（URL、cookie、TCP 连接建立时间戳、下载资源标识 ID）缓存于资源下载 HTTP 话单队列。

其次，计算下载资源 HTTP 话单的下载资源标识 ID，本文采用了累计哈希算法，计算一个 64 字节的字符串作为下载资源的标识 ID，算法伪代码如下：

算法 1：累计哈希算法

```
1)：Input: resource size; key; total accumulation length;
2)：Output: resource ID;
3)：for i:=0->size
4)：        hashL =hashL ^key[i]*BASE1;
5)：        hashH=hashH ^key[i]*BASE2;
6)：        accumulation length++;
7)：        resource ID=hashH<<32|hashL;
8)：end for
9)：if accumulation length==total accumulation length:
10)：       return resource ID;
11)：end if
```

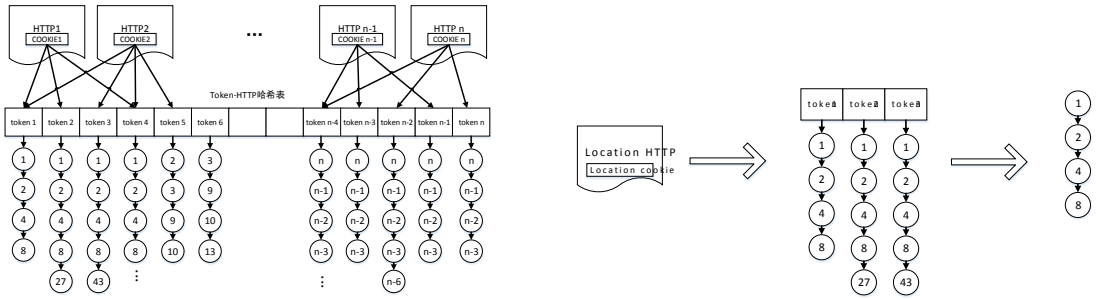
4.2　HTTP 话单索引模块

该模块对 HTTP 话单队列中出队的 HTTP 话单建立索引，规则如下：

（1）如果 HTTP 话单中存在 location 字段，则以 location 字段指定的 URL 作为 key，http 话单作为 value，存入 location-HTTP 话单哈希表；为了降低空间开销，该哈希表对 location 只做一定时间缓存。

（2）若存在 cookie，则将 cookie 分割为 token，以 token 作为 key，包含此 token 的 http 话单

作为



(a) 建立 cookie 与 HTTP 会话索引　　(b) 获取 URL 跳转链

图 3 URL 跳转链获取过程图

value，缓存于 token-cookie 哈希表。如图 3(a)所示， HTTP 话单 1、2、4、8 中都包含了 token1，因此在 token-http 哈希表中，token1 关键字所对应的数据则为 HTTP 话单 1、2、4、8 所构成的链表。为了

降低算法的时间开销和空间开销。在建立 token-http 话单哈希表时会去除区分度不高的 token，例如存在于大部分的 HTTP 话单的 token。如果区分度大于阈值 dif_0。则将此 token 从 token-http 话单中删除。如图 3（a）包含 token 4 和 token n-4 的 HTTP 话单大于阈值 dif_0，因此已经不能准确的标识用户的 HTTP 请求记录，所以将此 token 删除。

HTTP 话单建立索引算法的伪代码如下：

算法 2： HTTP 话单建立索引算法

```
1):Input :HTTP
2):Output :location HTTP    hashtable ;token HTTP    hashtable
3):if HTTP exists location
4):         Location HTTP    hashtable
5):if HTTP exists cookie
6):         token[n]=splits cookie by ';'
7):for i:=0-n
8):         insert http in token[i]-HTTP chain
9):         if    dif>dif0
10):              delete token[i]
11):        else
12):              token HTTP hashtabe.insert(key=token[i],data=HTTP)
13):end for
```

4.3 获取下载资源 URL 跳转链模块

该模块的处理过程包括三步。

Step1:将从下载资源的 HTTP 话单队列中出队的下载资源的 HTTP 话单的 URL 作为 key，查找 location-HTTP 话单哈希表，获取重定向 HTTP 话单。

Step2:将重定向 HTTP 话单的 cookie 分割成 token，以 token 为 key，查找 token-HTTP 话单哈希表，获取所有包含这些 token 的 HTTP 话单。如图 3(b)所示，对于某下载资源 HTTP 话单对应的重定向 HTTP 话单为 location HTTP，其 cookie 为 location cookie ，分割此 cookie 为 token1、token2、token 3，获取三个 token 分别对应的 HTTP 话单链，本文将其定义为疑似 HTTP 话单链。

Step3:遍历疑似 HTTP 话单链，将 HTTP 话单作为 key，以其出现的频率为 value，建立哈希表，统计 HTTP 话单在疑似 HTTP 话单链中出现的频率。如果其频率大于指定关联度阈值 sim_0，即认为其属于下载资源的 URL 跳转链。如图 3(b)所示，如果阈值为 2，1 比较三条 HTTP 链，则 HTTP 1、2、3、4 话单即为 URL 跳转链。

获取 URL 跳转链算法的伪代码如下：

算法 3：获取 URL 跳转链算法

```
1):Input: download HTTP
2):Output:url chain
3):Location HTTP=location-HTTP hashtable.Search (key=download HTTP )
4):token[n]=Splite cookie of location HTTP by ";"
5):for i:=0->n
6):         HTTP chain[i]= token HTTP hashtable.search(key=token[i])
7):end for
8):for j:=0->i*n
9):         if exists HTTP[j] in HTTP frequency hashtable
10):              frequency++
11):        if frequency > sim0
```

12): nsert HTTP[j] in url chain

13):end for

14):return url chain

4.4 分享链接查找模块

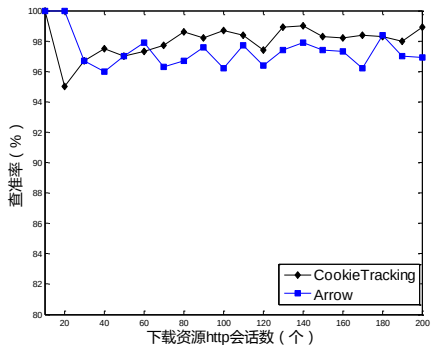
与从疑似 HTTP 话单中获取 URL 跳转链的方法类似，CookieID 统计的方法为：遍历相同资源标识 ID 的下载资源对应的多条 URL 跳转链的每个节点，建立以 URL 跳转链节点为 key，其出现频率作为 value 的哈希表，频率最大的 URL 即为分享链接。

4.5 分享链接验证模块

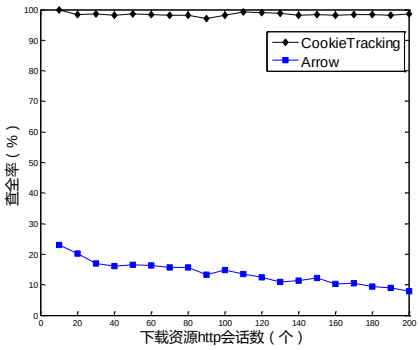
loadrunner 模拟用户访问分享链接，重新下载该资源，然后通过累计哈希计算该资源的标识 ID 值并与 CookieTracking 计算出的标识 ID 做对比，如果二者相同，则该资源的分享链接被确定。

5 实验和评价

下面验证 cookieID 方法的性能和可扩展性。首先给出评价指标的定义。查准率：指查找的正确分享链接所占查找的所有分享链接的比例。查全率：指查找的所有分享链接占所有下载资源数目的比例。获取时间：指获取分享链接的时间。扩展能力：随着流量的增长，查全率和正确率的变化。



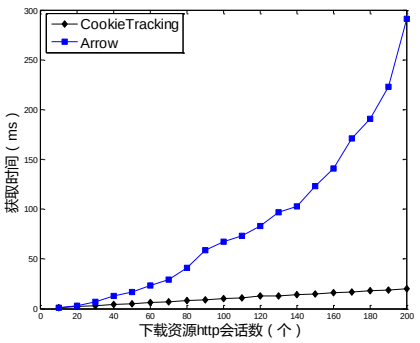
(a) 查准率



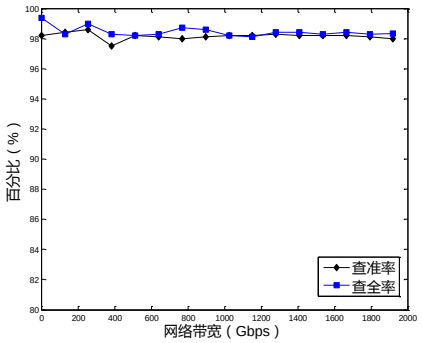
(b) 查全率

实验方法如下：将 CookieTracking 方法和 Arrow 方法[12]（通过离线学习分享链接 URL 生成的正则表达式去匹配获取网盘资源分享链接）同时在校园网同一网关上实验，首先，对比 CookieTracking 方法和 Arrow 方法的正确率、查全率、获取时间。然后，对 CookieTracking 方法进行压力测试，分析其~~拓展~~扩展能力。

实验结果为：如图 4(a) CookieTracking 方法和 Arrow 方法的查准率基本一致，平均查准率率分别是 98.67% 、 97.76% 。 如 图 4(b) 所 示 ， CookieTracking 的平均查全率为98.86%，而 Arrow 方法的查全率为 16.67%。由此可见，虽然 Arrow 方法具有和 CookieTracking 几乎相当的查准率，但是 Arrow 方法只能对 16.67%的网盘资源实现 URL 溯源，而 CookieTracking 方法可以实现 98.86%的网盘资源的 URL 溯源。



(c) 获取时间



（d） 查准率及查全率随带宽变化图

图 4 实验结果

如图 4(c) 说明随下载资源的增加，CookieTracking 的查找时间明显快于 Arrow，并且随着下载资源 HTTP 会话的增加，cookieID 方法的查找时间基本保持线程增长，而 Arrow 方法成指数增长。同时，如图 4(d)所示，压力测试结果表明：随着流量的增长，CookieTracking 方法的正确率和查全率基本保持不变。这证明 CookieTracking 方法具有很好的拓展扩展性。

6 结论

网盘资源分享链接作为目前互联网流行的资源传播方式，研究如何从骨干网络节点上的海量流量中识别出网盘下载 HTTP 会话的分享链接对于网络审查、网络取证、网络审计等具有重要意义。为此，本文提出一种高效可扩展的网盘资源分享链接获取方法——CookieTracking。CookieTracking 方法首先获取下载资源的 URL 跳转链，然后通过对比同一下载资源对应的不同 URL 跳转链获取唯一公共 URL 节点，认为该 URL 即为下载资源对应的分享链接。最后通过 loadrunner 模拟访问该 URL，验证正确性。实验结果表明：CookieTracking 方法具有很好的查准率、查全率、实时性、可拓展扩展性。下一步的工作是研究如何在更高带宽的骨干网络实现网盘资源分享链接的实时获取。

参考文献：

[1] Maier G, Feldmann A, Paxson V, et al. On dominant characteristics of residential broadband internet traffic[C]//Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. ACM, 2009: 90-102.

[2] Gehlen V, Finamore A, Mellia M, et al. Uncovering the big players of the web[M]. Springer Berlin Heidelberg, 2012.

[3] MobileTrends A. Global mobile broadband traffic report[R]. Allot Communications, Technical Report, 2010.[Online]. Available: http://www. allot. com/MobileTrends Report, 2010.

[4] Berghel H. The discipline of Internet forensics[J]. Communications of the ACM, 2003, 46(8): 15-20.

[5] Watts S, Newby J M, Mewton L, et al. A clinical audit of changes in suicide ideas with internet treatment for depression[J]. BMJ open, 2012, 2(5): e001558.

[6] Panah A, Panah A, Panah O, et al. Challenges of security issues in cloud computing layers[J]. Rep. Opin, 2012, 4(10): 25-29.

[7] Gokcen Y, Foroushani V A, Heywood A. Can we identify NAT behavior by analyzing Traffic Flows[C]//Security and Privacy Workshops (SPW), 2014 IEEE. IEEE, 2014: 132-139.

[8] Liu T T, Yang W, Xu C L, et al. A SNR-based Multi-channel Multicast Scheme for Popular Video in Wireless Networks[J]. Journal of Networks, 2013, 8(3): 628-635.

[9] Hayton S J, Jones D R, Lobo A R, et al. Using entity tags (etags) in a hierarchical http proxy cache to reduce network traffic: U.S. Patent Application 13/360,891[P]. 2012-1-30.

[10] Yen T F, Xie Y, Yu F, et al. Host Fingerprinting and Tracking on the Web: Privacy and Security Implications[C]//NDSS. 2012.

[11] Lee S, Kim J. Warningbird: A near real-time detection system for suspicious urls in twitter stream[J]. IEEE transactions on dependable and secure computing, 2013 (3): 183-195.

[12] Zhang J, Seifert C, Stokes J W, et al. Arrow: Generating signatures to detect drive-by downloads[C]//Proceedings of the 20th international conference on World wide web. ACM, 2011: 187-196.

[13] Jenefa A, Ravi R. Classifier: A Real-Time Detection system for suspicious URLs in Twitter Stream[J]. International Journal, 2014, 2(2).

[14] Aggarwal A, Rajadesingan A, Kumaraguru P. PhishAri: Automatic realtime phishing detection on twitter[C]//eCrime Researchers Summit (eCrime), 2012. IEEE, 2012: 1-12.

[15] Delosières L, Sánchez A. DroidCollector: A Honeyclient for Collecting and Classifying Android Applications[M]//Information Sciences and Systems 2014. Springer International Publishing, 2014: 175-183.

[16] Renkema-Padmos A, Volkamer M, Renaud K. Building castles in quicksand: blueprint for a crowdsourced study[C]//CHI'14 Extended Abstracts on Human Factors in Computing Systems. ACM, 2014: 643-652.

[17] Eshete B, Venkatakrishnan V N. WebWinnow: leveraging exploit kit workflows to detect malicious urls[C]//Proceedings of the 4th ACM conference on Data and application security and privacy. ACM, 2014: 305-312.

[18] Goldberg, Jeff, Magnus Westerlund, and Thomas Zeng. "A Network Address Translator (NAT) Traversal Mechanism for Media Controlled by Real-Time Streaming Protocol (RTSP)." (2014).

[19] Maier, Gregor, Fabian Schneider, and Anja Feldmann. "NAT usage in residential broadband networks." *Passive and Active Measurement*. Springer Berlin Heidelberg, 2011.

[20] Patel, Bhoomit, Jay Parikh, and Rushabh Shah. "A Review Paper on Comparison of SQL Performance Analyzer Tools: Apache JMeter and HP LoadRunner." (2014)