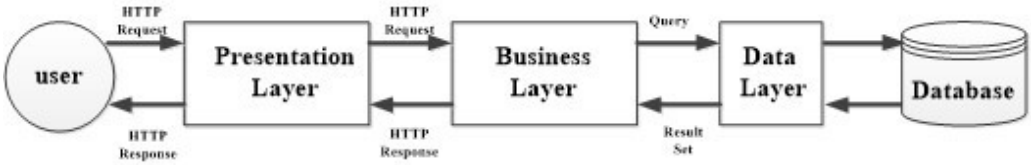

图1.网络应用程序的三层结构[8]。

流程。MiMoSA通过分析网络应用程序和数据库之间的关系，以及网络应用程序中的连接来检测多步骤攻击。

SENTINEL[11]和Pellegrino[3]采用了一种*预防方法*，以*黑箱的*形式识别业务逻辑漏洞。

SENTINEL[11]是一种*黑箱*方法，用于检测数据库访问中的逻辑弱点。SENTINEL生成一个网络的状态机，并从观察到的SQL查询和响应以及会话变量中提取一组不变式作为应用规范。任何违反定义的不变量的SQL查询都被识别为攻击。

Pellegrino等人。[3] 提出了一种*黑箱*技术来检测网络应用的逻辑漏洞。该技术从用户与某个应用的功能互动的网络痕迹中提取行为模式。首先，网络应用被建模，然后将攻击向量应用于模型。

我们以前的工作，BLDAST[12, 13]和BLTOCTTOU[14]使用预防方法，以黑箱方法的形式识别业务逻辑漏洞。BLProM[15]可以作为BLTOCTTOU和BLDAST的输入。

BLDAST[12, 13]是一个动态的、黑盒式的脆弱性分析方法，它可以识别网络应用程序的业务逻辑漏洞，以应对泛滥的DoS攻击。BLDAST评估网络应用程序对泛滥的DoS攻击的恢复能力。它可以考虑到网络应用的业务流程。BLDAST选择业务过程中的关键页面。一个关键的页面有相当长的响应时间。因此，一个关键的过程可以在目标中强制执行沉重的负载，并导致网络服务器变得没有反应。BLDAST的目标是在Web应用程序中找到这些关键进程。

BLTOCTTOU[14]是一个黑盒动态应用安全测试器，用于检测针对竞赛条件攻击的业务逻辑漏洞。BLTOCTTOU通过寻找网络应用程序的业务流程来识别漏洞。BLTOCT-TOU可以检测到相互作用的业务流程；一个流程应该设置一个变量的值，而另一个流程则应该设置另一个。

能够，而另一个应该读取或写入该变量。为了识别竞赛条件，BLTOCTTOU首先按顺序执行已识别的进程，然后按相反的顺序执行它们。最后，它评估了这两种模式的输出。如果它们是不同的，那么该网络应用程序就容易受到竞赛条件的影响。

2.2 对网页进行聚类

Crescenzi [16] 提出了一种基于页面结构的网页聚类方法。网页之间的结构相似性是由其超链接的DOM树定义的。最终的聚类被用来建立一个模型，根据网页的类别和它们的连接性来描述网站的结构。

3 业务层 过程中的矿工

在本文中，我们提出了BLProM来识别网络应用程序的业务流程，并将其输出作为网络应用程序业务层安全测试的输入。然后通过分析业务流程之间的互动，可以检测到业务层的漏洞。

BLProM首先对正常的用户HTTP流量进行预处理。然后提取流量中的网络应用页面。BLProM对类似的页面进行聚类，以防止用户导航图的无限增长。检测到的集群是图节点，图边是检测到的集群之间的关系。基于检测到的节点和边，用户导航图被提取出来。然后，BLProM从导航图中提取业务流程。BLProM有两个主要步骤。

1. 提取用户导航图。
2. 检测网络应用中的业务流程。

图2显示了在网络应用中识别业务流程的拟议步骤。在下文中，我们将详细解释这些步骤中的每一个。

3.1 提取用户导航图

首先，正常用户开始爬行网络应用程序。正常用户的流量被捕获和存储。应该注意的是，用户的权限

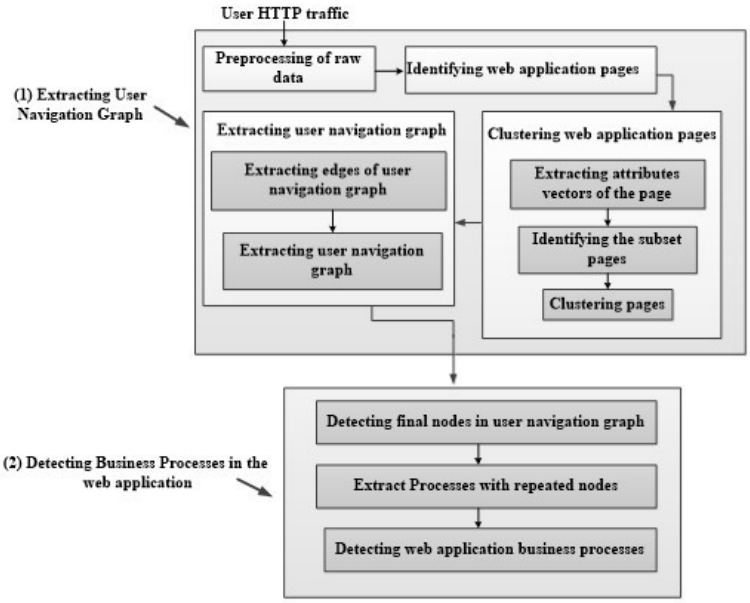

图2.检测网络应用程序业务流程的黑匣子方法。

级别可以根据用户的角色不同而不同，因此用户的导航图也因权限级别的不同而不同。在本文中，普通用户拥有用户级别的权限，并通过相关的允许的页面进行搜索。正常用户爬行允许搜索的应用程序的所有不同部分。

BLProM最初从存储的流量中提取用户导航图；这是通过以下步骤进行的。

1. 对原始输入数据进行预处理
2. 识别存储流量中的现有网络应用程序页面
3. 对网络应用程序页面进行分类
4. 提取用户导航图

1) *对原始输入数据进行预处理*
在预处理步骤中，BLProM对数据进行清理，并删除不相关的数据样本。在本文中，只使用了HTTP请求和响应。对于响应，只采用那些具有成功状态代码200和209的响应。BLProM删除了具有失败状态代码的响应，以及它们相应的请求。此外，在本文中，只需要GET和POST请求，其余的都被丢弃了。

2) *识别存储流量中的网络应用程序页面*

2) *识别存储流量中的网络应用程序页面*
应用程序的每个页面可以表示为一对（主请求，对主请求的相应响应）。为了识别网络应用的

如果要对这些页面进行分析，首先需要检测流量中的主要请求。在确定了主要请求后，还必须确定相应的响应。

识别流量中的主要HTTP请求。在用户的存储流量中，既有导致加载网络应用程序页面的主要HTTP请求，也有负责加载文件、图像等页面的次要HTTP请求。BLProM必须区分主请求和辅助请求。换句话说，当主要请求被确定后，其余的请求被认为是次要请求。
请求。这样，任何请求如果其引用者头与前一个请求的引用者不同，则被认为是次要请求，而前一个请求是主要请求。另外，流量中的第一个请求被认为是主请求，因为第一个请求不包括引用者。应该注意的是，HTTP请求中的Referer头域表示用户访问的前一个页面的URL。

识别**对主要请求的相应响应。**为了识别主请求的相应响应，只需要选择其内容类型字段为text/html的响应。因为次要请求的对应响应通常是文件、照片等，而主要请求的对应响应是文本和HTML的形式。这样的响应是主要响应，在
的HTTP流量。在确定了流量中的主要再请求和响应后，每一对（主要请求，对主要再请求的相应响应）都表示网络应用的一个页面。它

# Request URI	Referer	Response Content-Type
1 /osCommerce	/	text/html
2 / index.php?cPath=1	/osCommerce	text/html
3 /images/cdrm_drives.gif	/index.php?cPath=1	image/jpeg
4 / images/graphic_card.gif	/index.php?cPath=1	image/jpeg
5 /images/keyboards.gif	/index.php?cPath=1	image/jpeg
6 /index.php?cPath=1_17	/index.php?cPath=1	text/html
7 /css/stylings.php	/index.php?cPath=1_17	text/css
8 /index.php?cPath=1_4	/index.php?cPath=1_17	text/html

图3.连续请求的数量。

应该注意的是，在对主要的再请求的回应中，存在所有导致加载网页的次要请求。

识别流量中的最后一个请求是否是主请求。如果流量中的最后一个响应是主响应，那么最后一个HTTP请求也是主请求。

例如，在图3中，请求1、2、6和8是用红色显示的主要请求。

BL- ProM用于识别网络应用程序页面的算法的伪代码显示在算法1中。如前所述，每个网络应用程序页面都可以用一对（主请求，对主重寻的相应响应）来表示。算法1中的伪代码可以分为四个主要部分。

1. 提取流量中的主要请求（第13-26行）。
2. 提取对主要请求的相应响应（第27-33行）。
3. 识别路径中的最后一个请求是否是主请求（第34-37行）。
4. 提取网络应用程序页面（第38-43行） 在第9行，首先，流量中的所有请求，无论是主请求还是次要请求，都被提取出来，放在*HTTPRequest*变量中。在第10行，流量中的现有响应被添加到*HTTPResponse*变量中。在第11行和第12行，计算出请求和响应的总数。在第15行，检查当前的请求是否是第一个请求，如果是，它被认为是主要的请求并包括在*MainRequest*变量中。在第21行，检查当前请求的*Ref-erer*字段是否与前一个请求的Referer字段不同，如果是，前一个请求被认为是主要请求。在第29行，检查当前响应的content- type字段是否为text/html。如果满足给定的条件，当前的请求是被认为是主要的要求。

在第35行，检查最后的HTTP响应是否是主响应，如果是，最后的重新请求也被认为是主请求。

在第41行，所有主要的请求和响应都是网络应用程序的页面，它们被显示为成对的（请求，响应）。

3) *对网络应用程序页面进行分类*

在这一步，BLProM对网络应用程序页面进行聚类。聚类的目的是把类似的页面放在同一个聚类中。这有助于防止用户导航图的无限增长。同一聚类中的页面是彼此相似的。

在这个阶段，用户存储流量中的所有页面都被提取为一对（主请求，对主请求的相应响应）。每一对（主请求，对主请求的相应响应）都显示了应用程序的一个页面。为了提取最佳的用户导航图，必须对提取的类似页面进行识别和归类。在用户导航图中，节点表示应用程序的独特页面，边则代表页面之间的联系。为了识别类似的页面，聚类的目的应该考虑一个标准。用户可以在页面上进行的操作类型被认为是衡量页面分离的一个标准。换句话说，如果用户可以对其进行相同的操作，那么两个页面就是相似的。例如，考虑两个页面，它们都只包含一个按钮，但按钮的标题不同，第一个页面的标题是 "继续"，第二个页面的标题是 "保存"。这两个页面是不同的，因为用户对它们进行了不同的操作。因此，根据类似页面的标准，以下页面被认为是网络应用中的类似页面。

- 在网上商店的应用中，如果与商品的购物车有关的页面，即使它们包含不同的项目，它们也被认为是类似的页面。
- 在网上商店应用程序中，每个产品的简介页面都是相似的。因为它们在重要的HTML元素方面具有相同的HTML结构。
- 显示不同关键词搜索结果的页面是相似的。
- 如果页面的结构在重要的HTML元素方面是另一个页面的子集，那么这些页面被认为是相似的。
- 两个都包含用户评论的页面被认为是相似的，即使内容是关于不同的产品。

算法1 从流量中提取网络应用程序页面的伪代码

输入： HITRtraffic:{*HttpMessage* 1, *HttpMessage* 2, *HttpMessages* 3, . . ., *HithMessage* n}

OUTPUT。 网页是一组(*Request* i, *Respense* i)。

```
1: 开始
2: let JWebPages = {}; // 网页的集合
3：让 {}HttpRequest=; 我/HTTP请求的集合
4: Iet HittpResponse = {}; //一套HTTP响应。
5: let MainRequest= {}; // 主要HTTP请求的集合
6: let MainResponse = {}; // 主要HTTP响应的集合
7: let i, k= 1; // 当前HttpMessage的计数器。
8: let LastRefer = ∅, NewRefere =
9: HttpRequest=ExtractReg(HITRtraffic);//从HTTP流量中提取HTTP请求 10:
HttpRespense - ExtractResp(HITRtraffic);//从HTTP流量中提取HTTP响应 11: n =
extractNumber(HttpRequest) //提取HTTP请求的总数
12: m = extractNumber(HttpResponse) //提取HTTP响应的总数
13: //从HTTPRequests中提取主HTTP请求
14: for i: 1 ... n do
15:     如果（i = 1）那么
16:         add MainRequest ← HttpRequest i// 第一个请求是一个主请求
17:         HttpRequest的iLastReferRefer。
18:     否则
19:         HttpRequest的1NewRefereRefer。
20:     结束 如果
21:     如果(NewRefer /= LastRefer)那么22:
        添加MainRequestHttpRequesti-1; 23:
        LastReferNewRefere;
24:     结束 如果
25: 结束
26: //从HTTPResponse中提取主要HTTP响应
27: for k : 1 ... m do
28:     如果（HTTPResponse的k内容类型=text/html），那么
29:                         ← addMainRespenseHttpRespense k
30:     结束 如果
31: 结束
32: // 检查最后一个请求是否是主请求 33: 如果
(HTTPResponsem isinMainResponse) 则 34:
        addMainRequestHTTPRequest m
35: 结束 如果
36: //提取一组网页
37: size = extractNumber(MainRequest) //extract total number of Main HTTP Requests.
38: 对于j: 1 .....大小做
39:     WebPages ← (MainRequestj )MainResponsej)。
40: 结束
41: 返回WebRages。
42：结束
```

表1.osComerce网络应用程序中页面的属性向量。

投入	无
钮扣	html.body.button.Review# html.body.div.div.form.div.span.span.button.Add to Cart
锚	无
形象	html.body.div.div.a.img

```
<html>
        <title> Hello! </title>
        <body>
            <p>
                <button value= “continue”> Continue</button>
            </p>
        </body>
</html>
```

图4.一个HTML代码的例子。

一个HTML元素的文档对象模型（DOM）路径的定义。一个元素的DOM路径是指该元素在HTML代码中的位置。

例如，在图4中，按钮的DOM路径（*DOMbutton*）是Document.Html.Body.P.Button。

定义1.【类似的页面】类似的页面是指用户可以对其进行相同的操作，并且在页面中重要的HTML元素的位置方面是相同的。页面中的重要HTML元素包括按钮、图片、输入和锚。

聚类过程包括三个步骤。

1. 提取页面的属性向量
2. 识别子集页面
3. 聚集的页面

在下文中，将详细讨论这些步骤。

1. **提取页面的属性向量** BLProM将应用程序的每个页面显示为一对（主请求，响应）。在这一步中，BLProM通过对上述一对数据进行最小化操作来提取每个页面的相应属性向量。BLProM使用以下属性向量对每个页面进行建模。

WebPages = 一个应用程序中的总页数

$\forall$ w e 网页

w= (DOM_{inputs}, $DOM_{buttons}$, $DOM_{anchors}$, DOM_{imgs})

$DOMinputs = {}_{i}^{n}DOM(input_i)$

$DOMbuttons = {}_{i}^{n}DOM(按钮_i)$

$DOManchors = {}_{i}^{n}DOM(anchor_i)$

$DOM_{印象} = {}_{i}^{n}DOM(img_i)$

- DOM(input): 页面中<*输入*>标签的DOM路径+<*输入*>标签中type属性的值+<*输入*>标签中name属性的值（如果没有name属性，则考虑value属性）。
- DOM(button)。页面中按钮的DOM路径+按钮的标题。
- DOM(anchor)。页面中< *a* >标签的DOM路径。
- DOM(img)。页面中现有图像的DOM路径。

假设网络应用程序页面包含几个按钮，在这种情况下，页面属性向量的第二个元素是页面中按钮的DOM路径的集合，这些路径用 "#"分隔。图5显示了其中一个osCommerce[1]网络应用程序页面。表1显示了图5中页面的at-tribute向量。如图所示，输入元素和锚点元素为空，这意味着该页面不包含上述标签。

2. **识别类似页面**

在提取了每个页面的属性向量后，有必要识别类似的页面。那些其属性向量是另一个页面的子集或具有完全相似的属性向量的页面被认为是相似的页面。根据定义1，每个网络应用程序页面的属性向量有四个元素。第1页的属性向量被认为与第2页的属性向量相同

如果。

- 第1页的所有向量元素等于第2页向量中的响应元素。
- 第1页的所有向量元素是第2页向量中相应元素的子集。
- 第2页的所有向量元素是第2页向量中相应元素的一个子集。

第1页

。

[1]https://www.oscommerce.com/

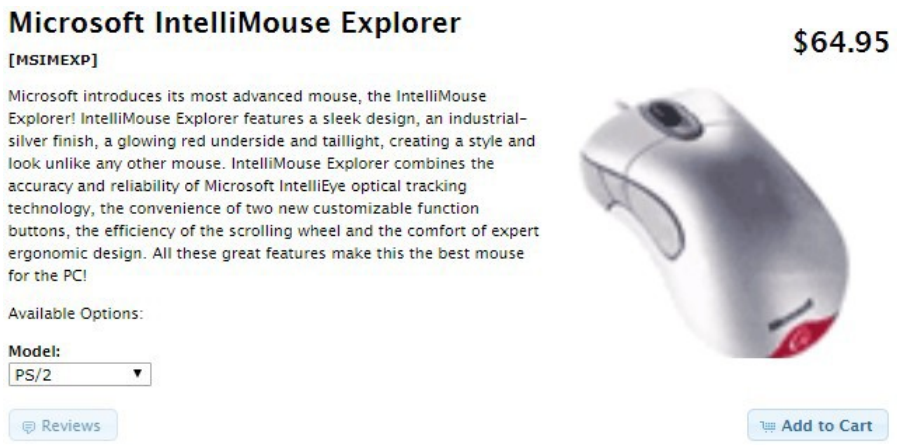

图5.osCommerce应用程序的一个页面。

算法2 识别相似页面的伪代码

输入： $w_1 = (DoM_{w1}(input), DoM_{w1}(button), DoM_{w1}(anchor), DoM_{w1}(img))$,
$w_2 = (DoM_{w2}(input), DoM_{w2}(button), DoM_{w2}(anchor), DoM_{w2}(img))$
OUTPUT: 布尔标志 // 真意味着两个页面是相同的

```
1: 开始
2：让flag、input、button、anchor、img=false。
3：如果DoM_w1（输入）⊆ DoM_w2（输入）或DoM_w2（输入）
                DoM_w1（输入）则
4:      输入=true。
5: 结束 如果
6：如果DoM_w1（按钮）⊆ DoM_w2（按钮） 或DoM_w2（按钮）
                DoM_w1（按钮），那么
7:      button=true。
8: 结束 如果
9：如果DoM_w1（锚）⊆ DoM_w2（锚）或DoM_w2（锚） DoM_w1（锚）则
10:     anchor=true。
11: 结束 如果
12: 如果 DoM_w1(img)⊆ DoM_w2(img) 或 DoM_w2(img) DoM_w1(img) 则
13:     img=true;
14: 结束 如果
15：如果输入和按钮以及锚和IMG，那么
16:     flag=true。
17: 结束 如果
18：返回标志。
19：结束
```

- 如果第1页的一个或多个向量元素是第2页向量中相应元素的一个子集，那么第1页的其余向量元素必须与第2页向量中的相应元素相同。
- 空元素是每个元素的一个子集。

类似的网页是根据上述属性来识别的。算法2说明了识别类似页面的伪代码。

3. **对网络应用程序页面进行聚类**

在确定了类似的页面后，它们是放在同一个集群中。一个群组中的网页彼此相似，并且指的是应用程序的一个独特的页面。算法3显示了对网页进行聚类的伪代码。在第7行，检查两个网页wi和wj是否相同，如果是，就把它们放在同一个簇中。

4) *提取用户导航图*

在这一步，BLProM将获得的集群连接起来，每个集群代表一个独特的网络应用页面。每个集群都有一组类似的页面，每个页面都有URI和参考者字段。因此，每个集群都包含一组URI和一组该集群中的页面的引用者。应该注意的是，Referer字段是URI

算法3 对网络应用程序页面进行聚类的伪代码

INPUT: WebPages = $w\{\}w_2, w_3, \ldots, w_n$

输出：网络应用程序模型M作为一组网页群集C；网页群集C作为一组网页。

```
1: 开始
2: 让M = ∅; // 页面集群的集合
3: 让k=1; //网页集群的数量
4: for i : 1 ... n do
5:     C k← w i
6:     for j = i + 1 ... n do
7:         如果SimilarWebPages(w i, w j) 那么
8:             WebPages-WebPagesw;
9:             n=网页的长度。
10:            C k← w j
11:        结束 如果
12:    结束时间
13:    k + +;
14: 结束
15：返回C。
16：结束
```

用户访问的前一个网络应用程序的页面。为了提取用户导航图的边，对产生的聚类进行检查，找出哪个聚类的Reference集合与聚类的URI集合有交集。当集群被找到时，这两个集群就被连接起来。假设集群C1的URI集与集群C2的Refer集有交集，那么从集群C1到集群C2的路径 （C1C2）就被创建。换句话说，边C1C2是用户导航图中的一条边。算法4显示了从用户导航图中提取边的伪代码。根据算法3中的伪代码第5行和第6行，分别得到每个集群的URI集和Referer集。在第10行，如果每个集群的URI集与其他集群的Referer集的交集不是空的，CiCj边就被添加到图的边集中。在这一步中，BLProM将获得的集群连接起来，每个集群代表一个独特的网络应用页面。每个集群实际上都有一组类似的网页，这些网页都有URI和Referer字段。因此，每个集群都包含一组URI和一组集群中的页面的参考者。应该注意的是，Referer字段实际上是用户访问的前一个网络应用程序页面的URI。为了提取用户导航图的边，对产生的集群进行检查，找出集群中的哪个Referer集与该集群的URI集有交集，当找到该集群时，这两个集群是相连的。假设集群C1的URI集与集群C2的Refer集有交集，那么从集群C1到集群C2的路径（C1C2）被创建。换句话说，边C1C2是用户导航图中的一条边。算法4显示了从用户导航图中提取边的伪代码。根据算法3中的第5行和第6行，分别得到每个集群的URI集和Referer集。在第10行，如果每个集群的URI集与其他集群的Referer集的交集不是空的，CiCj边就被添加到图的边集。用户导航图是根据算法5创建的，其中节点是集群集，确定的边是集群之间的路径。在创建的图中，每个节点表示唯一的页面，边表示页面之间的路径。

定义2.[用户导航图]该图用元组< C0, C, E>表示，其中C是图中的节点集合，C0 C是图中的第一个（初始）节点，E C C是图中的边的集合。

算法5显示了提取用户导航图的伪代码。第7行表示一个包含应用程序的初始页面的集群。它被认为是图的初始节点。

3.2 识别应用中的业务流程

为了确定用户导航图中的网络应用程序业务流程，有必要定义流程和最终节点，然后定义业务流程。

定义3.[应用过程（P）]。

算法4 从用户导航图中提取边的伪代码

INPUT:C = {$C_1, C_2, C_3, \ldots, C_k$}//网页集群作为图形节点 WebPages= $w_1, w_2, w_3, \ldots, w_n$//网页的集合

输出：网络应用程序图的边缘E是一个边缘集合

1: 开始
2: 让E = ∅; // 网络应用图边的集合
3: **对于***j* : 1 . . . *k***做**
4: 让URI_{cj}, $Referer_{cj}$= ∅ 。
5: URI_{cj}= URI_{cj} ∪ ExtractURI (w) for any w∈C_j
6: $Referer_{ck}$= $Referer_{ck}$ ∪ ExtractReferer(w) for any $w \in C_j$
7: **结束**
8: **for** i: ... *k* **do**
9: **for** *j* : *i* + 1 ... *k* **do**
10: **如果** (URI_{ci} ∩_∩) $Referer_{cj}$ /= *null*) **则**
11: $E \leftarrow C_i C_j$
12: **结束 如果**
13: **结束时间**
14: **结束**
15：返回E。
16：结束

算法5 提取用户导航图的伪代码 **INPUT:**C= {$C_1, C_2, C_3, \ldots, C_k$}//网页集群作为图的节点 w_1//第一个网页

输出：网络应用程序导航图< C_0, C, E >。

1: 开始
2: 让C_0= ∅
3: E=ExtractGraphEdges; //网络应用程序图边的集合
4: **对于***j* : 1 . . . *k***做**
5: **如果***Ck*包含w_1，**那么**
6: $C_0 = C_o + C_k$
7: **结束 如果**
8: **结束**
9：返回C_0 ，C ，E。
10：结束

应用程序中的过程P是用户导航图中的节点和边的序列，如$E_1, E_2, \ldots, E_k$，*其中*$E_i \in E$，*并且*$E_i = C_{i-1}C_i$。

算法6显示了提取进程的伪代码。

定义4.[用户导航图（F）中的最终节点] 最终节点指的是应用程序到达那里时发生的业务流程的完成。

最后的节点可以通过检查HTTP响应来检测。例如，在购买产品的过程中，完成购买后会显示 "感谢您的购买 "这样的短语。通过指定一组这些短语并在响应中搜索它们，最终节点可以被识别出来。

fied。一些用于识别最终节点的关键词是谢谢、祝贺、成功、注销和搜索结果。此外，网络应用程序页面中的一些按钮是很好的标志，有助于识别图中的最终节点。最终节点的例子包括点击 "保存 "按钮后的页面、点击 "创建 "按钮后的页面和点击 "提交 "按钮后的页面。

定义5.[应用程序的业务流程]应用程序中的业务流程BP是一个至少具有下列条件之一的流程。

1. 流程的第一个节点是用户导航图（C_0 ）中的初始节点，流程结束节点是用户导航图（F ）中的最终节点。
2. 如果进程再次通过它的第一个节点，这意味着

算法6 提取过程的伪代码
输入：网络应用程序的第一个节点C_0
网络应用图边E
输出：网络应用程序图进程P作为网络应用程序的集合

```
1: 开始
2: 让P0= ∅; // 网络应用程序的集合
3：让StartEdge= ∅; //从C0开始的图边集。
4: StartEdge= ExtractGraphEdges(C0) //从C0中提取所有边。
5: EndPoint = ExtractEndPoint(StartEdge) //提取边缘的端点。
6：如果（ExtractProcess(EndPoint, E) = null），那么
7:      返回 P=E+ExteractProcess(Endpoint, E); 对于任何边  ∈ EStartEdge
8：否则
9:      返回E;
10: 结束 如果
11：结束
```

流程的第一个节点和终点节点是相同的，并且流程长度大于2。(如果在流程中有一个返回到通过的节点，并且创建的循环长度大于2，这就是一个业务流程）。

图中从初始节点（应用初始页）到确定的最终节点的所有进程，以及其初始节点和最终节点的进程都是相同的；并且所有这些都是应用业务流程。算法7显示了识别应用业务流程的伪代码。在第4行，应用程序的业务流程被提取出来。在第5行，提取图的最终节点。在第7行，以初始节点开始，以最终节点结束的流程被识别为业务流程，存储在变量BP中。在第9行，检测有重复节点的过程，在第11行，在检测到的过程中，如果它们的初始节点和最终节点是相同的，并且它们的长度大于2，它们被添加到变量BP中作为业务流程。

4 实验结果

本节使用的测试平台是一个网络，由一个网络服务器（测试目标）和两个客户端（BLProM系统和合法用户）组成。网络服务器和客户端被加载在一个虚拟机上。网络服务器和客户端的情况如表2所示。

表3中列出的网络应用程序被安装在网络服务器上（测试目标），然后我们计划确定网络应用程序的业务层。

合法用户首先开始使用选定的网络应用程序。该用户抓取网络应用的所有允许部分。合法用户的HTTP流量被提供给BLProM作为其输入。

5 评价

BLProM的目标是识别网络应用程序的业务层。我们可以通过识别网络应用程序的业务层来识别业务逻辑的脆弱性。BLProM检测网络应用程序的业务流程。识别业务流程是对业务层的Web应用进行动态安全测试的主要步骤。

我们将BLProM与OWASP ZAP进行比较。OWASP Zed Attack Proxy（ZAP）是一个免费的网络扫描器。它扫描网络应用并自动发现一些安全漏洞。ZAP是唯一的免费网络扫描器，它有提取网络应用程序图的API。网络应用程序页面是图的节点，页面之间的关系显示为图的边。BLPRoM和ZAP的主要区别在于检测类似的页面。ZAP不能检测网络应用中的类似页面，但BLPRoM可以。ZAP的图只显示了扫描页面之间的关系，但BLPRoM生成了最佳图。关于生成图形的准确性，BLProM和ZAP都是一样的。

为了评估所提出的方法，我们首先通过以下标准显示聚类的准确性。

- 真正的阳性。符合其正确群组的样本。
- 假阳性。符合某一聚类的样本，但不属于该聚类。
- 假阴性。不适合在一个聚类中的样本，但它们属于该聚类。
- 召回。它由以下公式计算。

$$召回= \frac{TruePositive}{TruePositive+FalseNegative}$$

- 精度。它是通过以下公式计算的。