Checking Security Properties of Cloud Service REST APIs

会议：IEEE ICST 13th(2020/10)，CCF C类

作者：来自Columbia University与Microsoft Research的研究团队

Presenter:zouyue

1 INTRODUCTION

- **Use-after-delete rule.**被删除的资源不可再次被访问
- **Resource-leak rule.**没能成功创建的资源不能被访问且不能泄露后端状态信息
- **Resource-hierarchy rule.**父资源的子资源不能从另一个父资源访问
- **User-namespace rule.**在用户命名空间中创建的资源不能从另一个用户命名空间访问

违反此类规则可能导致特权提升攻击，信息泄露攻击与DDOS攻击等

2 SECURITY CHECKERS FOR REST APIS

Use-After-Free Rule：

- 例子：发起删除/users/user-id1请求后，其他所有尝试访问user-id1的请求都必须失败且返回404.
- **Use-after-free checker.** 执行delete操作后触发主驱动程序

```
Inputs: seq, global_cache, reqCollection
# Retrieve the object types consumed by the last request and
# locally store the most recent object id of the last object type.
n = seq.length
req_obj_types = CONSUMES(seq[n])
# Only the id of the last object is kept, since this is the
# object actually deleted.
target_obj_type = req_obj_types[−1]
target_obj_id = global_cache[target_obj_type]
# Use the latest value of the deleted object and execute
# any request that type−checks.
for req in reqCollection:
    # Only consider requests that typecheck.
    if target_obj_type not in CONSUMES(req)
        continue
    # Restore id of deleted object.
    global_cache[target_obj_type] = target_obj_id
    # Execute request on deleted object.
    EXECUTE(req)
    assert ''HTTP status code is 4xx''
    if mode != 'exhaustive':
        break
```

Fig. 1: **Use-after-free checker.**

2 SECURITY CHECKERS FOR REST APIS

Resource-leak rule：

- 例子：在发出格式错误的 PUT 请求来创建 URI / users/user-id1 后，必须收到 4xx 响应。任何后续访问（读取、更新或删除）此 URI 的请求也必须失败，且不能透露后端信息。
- **Resource-leak checker.**收到无效状态码的响应后触发。

```
Inputs: seq, global_cache, reqCollection
# Retrieve the object types produced by the whole sequence and by
# the last request separately to perform type checking later on.
seq_obj_types = PRODUCES(seq)
target_obj_types = PRODUCES(seq[−1])
for target_obj_type in target_obj_types:
    for guessed_value in GUESS(target_obj_type):
        global_cache[target_obj_type] = guessed_value
        for req in reqCollection:
            # Skip consumers that don't consume the target type.
            if CONSUMES(req) != target_obj_type:
                continue
            # Skip requests that don't typecheck.
            if CONSUMES(req) − seq_obj_types:
                continue
            # Execute the request accessing the ''guessed'' object id.
            EXECUTE(req)
            assert ''HTTP status code in 4xx class''
            if mode != 'exhaustive':
                break
```

Fig. 2: **Resource-leak checker.**

2 SECURITY CHECKERS FOR REST APIS

Resource-hierarchy rule:

- 例子：向/users/user-id1、/users/user-id2 和 /users/user-id1/reports/report-id1 发出 POST 请求创建用户 user-id1、user-id2，然后将 report-id1添加给用户 user-id1，对/users/user-id2/reports/report-id1 的后续请求必须失败，因为根据资源层次结构规则。
- **Resource-hierarchy checker.**替换父对象的id进行测试。

```
1  Inputs: seq, global_cache
2  # Record the object types consumed by the last request
3  # as well as those of all predecessor requests.
4  n = seq.length
5  last_request = seq[n]
6  target_obj_types = CONSUMES(seq[n])
7  predecessor_obj_types = CONSUMES(seq[:n])
8  # Retrieve the most recent id of each child object consumed
9  # only by the last request. These are the objects whose
10 # hierarchy we will try to violate.
11 local_cache = {}
12 for obj_type in target_obj_types − predecessor_obj_types:
13   local_cache[obj_type] = global_cache[obj_type]
14 # Render sequence up to before the last request
15 EXECUTE(seq, n−1)
16 # Restore old children object ids that do NOT belong to
17 # the current parent ids and must NOT be accessible from those.
18 for obj_type in local_cache:
19     global_cache[obj_type] = local_cache[obj_type]
20 EXECUTE(last_request)
21 assert ''HTTP status code is 4xx''
```

Fig. 3: **Resource-hierarchy checker.**

2 SECURITY CHECKERS FOR REST APIS

User-namespace rule：

- 例子：在发出 POST 请求以使用令牌 token-of-user-id1 创建 URI /users/user-id1 后，资源 user-id1 不得使用另一个用户的另一个令牌 token-of-user-id2 进行访问
- **User-namespace checker.**尝试使用不同的身份验证令牌重新执行主驱动程序执行的任何测试用例的有效最后请求。

2 SECURITY CHECKERS FOR REST APIS

Combining All Checkers

每当有状态 REST API 模糊器达到新状态，执行图示的代码，四个主动检查器不会冲突。

```
1  Inputs: seq, global_cache, reqCollection
2  # Execute the checkers after the main driver.
3  n = seq.length
4  if seq[n].http_type == ''DELETE'':
5          UseAfterFreeChecker(seq, global_cache, reqCollection)
6  else:
7  if seq[n].http_response == ''4xx'':
8          ResourceLeakChecker(seq, global_cache, reqCollection)
9  else:
10         ResourceHierarchyChecker(seq, global_cache)
11         UserNamespaceChecker(seq, global_cache)
```

Fig. 4: **Checkers dispatcher.**

3 EXPERIMENTAL EVALUATION

实验设置

对三个云服务进行实验，Azure A 和 Azure B 是两个Azure 管理服务，O-365 C是一个 Office365 消息服务，请求数量范围为 13~19 个，测试时间为1小时。

API	Total Req.	Mode	Statistics		Bug Buckets				
			Tests	Checkers	Main	Use-Aft-Free	Leak	Hierarchy	NameSpace
Azure A	13	optimized	4050	45.0%	4	3	0	0	0
		exhaustive	2174	54.5%	4	3	0	0	0
Azure B	19	optimized	7979	46.2%	0	0	1	0	0
		exhaustive	9031	63.9%	0	0	1	0	0
O-365 C	18	optimized	10982	4.1%	1	0	0	1	0
		exhaustive	11724	11.4%	0	0	0	1	0

TABLE II: **Comparison of modes *optimized* and *exhaustive* for two Azure and one Office-365 services.** Shows the number of requests sent in 1 hour (Tests) with BFS-Cheap, the percentage of tests generated by all four checkers combined (Checkers), and the number of bug buckets found by the main driver and each of the four checkers. *Optimized* finds all the bugs found by *exhaustive* but its main driver explores more states faster given a fixed test budget (1 hour).

2 SECURITY CHECKERS FOR REST APIS

Resource-leak violation in Azure：

- **例子**：1.创建（**put**）类型为 CM、名称为 X 且具有特定错误格式主体的新资源，返回500错误，创建失败。 2.请求**(get)**所有CM类型资源的信息，返回空列表。 3.在不同地区再次创建一个类型为 CM、名称为 X 正确格式的对象

- **结果分析**：最终返回**409 Conflict**错误，存在潜在的安全风险。

Resource-hierarchy violation in Office365:

- **例子**：1.创建一条消息msg-1(request :POST /api/posts/msg-1). 2. 创建第二天消息msg-2 (request:POST /api/posts/msg-2). 3.为msg-1创建一条回复reply-1(request: POST /api/posts/msg-1/replies/reply-1). 4.在msg2下申请编辑reply-1(request:PUT /api/posts/msg-2/replies/reply-1)
- **结果分析**：服务器同意了请求，说明服务器缺少层次结构验证检查，是潜在的安全漏洞。

THANKS