Building a defense strategy for your Cloud workloads

Henrik Johansson
Principal SA Content PM, AWS Security
What to expect

Secure architecture flow
Attack sources/types
Key items in building cloud security
Q&A
Secure architecture flow
Secure architecture flow
Let’s simplify
Your stuff
Your stuff
All your (code)base are belong to us

Your stuff
All your (code)base are belong to us
What are we protecting against?
External threats
External threats

- Trojans
- "Hackers"
- Script kiddies
- Hacktivists
- Researchers?
- State-Sponsored Attackers
Internal threats
Internal threats

Disgruntled employees

Financially driven

Oops...
Internal threats

Disgruntled employees

Financial driven

Oops...
What about **DDOS**?
What about DDOS?

DDOS

- Reflection and amplification
- Layer 4 and 7 floods
- Slowloris
- SSL abuse

HTTP floods

- SQL injection
- Bots and probes
- Application exploits

Targeted attacks

- Social engineering
- Reverse engineering

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
HOW?
First…
# AWS Shared Responsibility Model

## HARDWARE/AWS GLOBAL INFRASTRUCTURE

<table>
<thead>
<tr>
<th>Regions</th>
<th>Availability Zones</th>
<th>Edge Locations</th>
</tr>
</thead>
</table>

## SOFTWARE

<table>
<thead>
<tr>
<th>Compute</th>
<th>Storage</th>
<th>Database</th>
<th>Networking</th>
</tr>
</thead>
</table>

## CUSTOMER DATA

<table>
<thead>
<tr>
<th>Platform, Applications, Identity &amp; Access Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System, Network &amp; Firewall Configuration</td>
</tr>
</tbody>
</table>

## AWS

- Responsibility for security ‘of’ the cloud

## CUSTOMER

- Responsibility for security ‘in’ the cloud
  - Client-Side Data Encryption & Data Integrity Authentication
  - Server-Side Encryption (File System and/or Data)
  - Networking Traffic Protection (Encryption, Integrity, Identity)

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Use it in your defense strategy

https://aws.amazon.com/compliance/shared-responsibility-model/
Key items in building a cloud defense strategy
(No, this is not a checklist)
Organization
Meet the new security team

Operations  Engineering

Application Security  Compliance
Meet the new security team

- Operations
- Engineering
- Development
- Application Security
- Compliance
Visibility
VISIBILITY

HOW OFTEN DO YOU MAP YOUR NETWORK?

WHAT’S IN YOUR ENVIRONMENT RIGHT NOW?
Remember

There is no server under the table...
Security is **Visible**

Who is accessing the resources?
Who took what action?
- When?
- From where?
- What did they do?
- Logs Logs Logs

_Understand what is available, and who should get it_
Security is **Visible**

Who is accessing the resources?
Who took what action?
- When?
- From where?
- What did they do?
- Logs Logs Logs

**Understand what is available, and who should get it**

With great visibility comes great possibilities
Example: VPC Flow Logs – See all your traffic

- Agentless
- Enable per ENI, per subnet, or per VPC
- Logged to AWS CloudWatch Logs
- Create CloudWatch metrics from log data
- Alarm on those metrics
What is happening in your account

```json
{
  "detail-type": [ "AWS API Call via CloudTrail" ],
  "detail": {
    "eventSource": [ "cloudtrail.amazonaws.com" ],
    "eventName": [ "StopLogging" ]
  }
}
```
#!/bin/bash
REGION=$(wget -q -O - http://169.254.169.254/latest/meta-data/placement/availability-zone|sed 's/[^\{\}]*\{1\}\}/\}/g')
DATE=$(date)
aws ec2 --region $REGION create-tags --resources $INSTANCE_ID --tags "Key=Tainted,Value=$DATE"

{
"detail-type": [ "AWS API Call via CloudTrail" ],
"detail": {
  "eventSource": [ "ec2.amazonaws.com" ],
  "eventName": [ "CreateTags" ],
  "errorCode": [ "Client.UnauthorizedOperation" ]
}

Adapt visibility to situation

```python
def buildMessage(data, isRoot):
    # Create message based on situation
    message = "=-=-=-=-= User activity tracker alert ==========
"
    message = message + "== Only Read/Write events tracked ==
"
    if isRoot is True:
        message = message + "DEFCON 5 WARNING! Root is being used!
"
    message = message + "Event time: " + data['eventTime'] + "\n"
    message = message + "Account: " + data['account'] + "\n"
    message = message + "Region: " + data['region'] + "\n"
    message = message + "\n"
    message = message + "UserIdentity: \n"
    for n, m in data['userIdentity'].items():
        message = message + "\t" + n + "": " + m + "\n"
    message = message + "\n"
    message = message + "Source info: \n"
    message = message + "User Agent: " + data['userAgent'] + "\n"
    message = message + "Source IP: " + data['sourceIP'] + "\n"
    message = message + "\n"
    message = message + "API: " + data['eventName'] + "\n"
    message = message + "API request parameters:\n"
    for n, m in data['requestParameters'].items():
        message = message + "\t" + n + "": " + m + "\n"
    message = message + "\n"
    if data['errorCode'] != "":
        message = message + "Error messages: \n"
        message = message + "Error Code: " + data['errorCode'] + "\n"
        message = message + "Error Message: " + data['errorMessage'] + "\n"
    message = message + "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
"
    return message
```
Expand visibility

Slack

Amazon Chime

Email using Amazon SNS
Data flow
Data flow

What is the expected flow of traffic?

Can it take other paths?
Understand your data flow
Ensure strict data flow
Example: Edge protection

How do you handle scaled attacks
Types of DDoS attacks

**Application-layer DDoS attacks**
Use well-formed but malicious requests to circumvent mitigation and consume application resources (e.g., HTTP GET, DNS query floods)

**State-exhaustion DDoS attacks**
Abuse protocols to stress systems like firewalls, IPS, or load balancers (e.g., TCP SYN flood)

**Volumetric DDoS attacks**
Congest networks by flooding them with more traffic than they are able to handle (e.g., UDP reflection attacks)
Challenges in mitigating DDoS attacks

- Complex set-up
- Provision bandwidth capacity
- Application re-architecture
DDoS protections built into AWS

✓ Protection against most common infrastructure attacks

✓ SYN/ACK Floods, UDP Floods, Reflection attacks etc.

✓ No additional cost
Example: Layer control

Control access

Reduce scope

Integrate in threat model
Security Groups = stateful firewall

In English: Hosts in this group are reachable from the Internet on port 80 (HTTP)

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP (80)</td>
<td>TCP (6)</td>
<td>80</td>
<td>0.0.0.0/0</td>
</tr>
</tbody>
</table>
Security Groups = stateful firewall

In English: Hosts in this group are reachable from the Internet on port 80 (HTTP)

Default = Deny

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP (80)</td>
<td>TCP (6)</td>
<td>80</td>
<td>0.0.0.0/0</td>
</tr>
</tbody>
</table>
Not just one way – Prevent collateral damage
Example: Access control

Who can access?

From where?

How?
Access control

Use existing if possible/suitable

Reduce access
   Remember, all access is logged

Easy to temporarily revoke

Don’t forget temporary keys
# IAM Credential Reports

Click the button to download a report that lists all your account’s users and the status of their various credentials. After a report is created, it is stored for 90 days. For information on how often reports are created, see the **Reports** section in the **IAM User Management** console.

<table>
<thead>
<tr>
<th>User</th>
<th>ARN</th>
<th>User creation date</th>
<th>Password status</th>
<th>Last used date</th>
<th>Password status</th>
<th>MFA activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>root_account</td>
<td>arn:aws:us-east-1:root</td>
<td>2014-06-01</td>
<td>not_supported</td>
<td>2014-11-05T23:02:18+00:00</td>
<td>not_supported</td>
<td>TRUE</td>
</tr>
<tr>
<td>amacdermott</td>
<td>arn:aws:us-east-1:macdermott</td>
<td>2014-08-1</td>
<td>TRUE</td>
<td>2014-09-2</td>
<td>2014-11-1</td>
<td>FALSE</td>
</tr>
<tr>
<td>cwhalley</td>
<td>arn:aws:us-east-1:whalley</td>
<td>2014-08-1</td>
<td>TRUE</td>
<td>2014-08-1</td>
<td>2014-10-0</td>
<td>FALSE</td>
</tr>
<tr>
<td>lford</td>
<td>arn:aws:us-east-1:ford</td>
<td>2014-08-1</td>
<td>TRUE</td>
<td>2014-09-2</td>
<td>2014-11-1</td>
<td>FALSE</td>
</tr>
<tr>
<td>ilegal</td>
<td>arn:aws:us-east-1:ilegal</td>
<td>2014-08-1</td>
<td>TRUE</td>
<td>2014-09-2</td>
<td>2014-11-1</td>
<td>FALSE</td>
</tr>
<tr>
<td>mbretan</td>
<td>arn:aws:us-east-1:mbretan</td>
<td>2014-10-1</td>
<td>TRUE</td>
<td>2014-10-22T17:27:25+00:00</td>
<td>2014-10-1</td>
<td>2014-12-0</td>
</tr>
<tr>
<td>pmalhotra</td>
<td>arn:aws:us-east-1:pmalhotra</td>
<td>2014-08-1</td>
<td>TRUE</td>
<td>2014-09-2</td>
<td>2014-11-1</td>
<td>FALSE</td>
</tr>
<tr>
<td>sdiffer</td>
<td>arn:aws:us-east-1:sdiffer</td>
<td>2014-08-1</td>
<td>TRUE</td>
<td>2014-09-2</td>
<td>2014-11-1</td>
<td>FALSE</td>
</tr>
<tr>
<td>stwaddle</td>
<td>arn:aws:us-east-1:stwaddle</td>
<td>2014-08-1</td>
<td>TRUE</td>
<td>2014-09-2</td>
<td>2014-11-1</td>
<td>FALSE</td>
</tr>
<tr>
<td>ttstrobell</td>
<td>arn:aws:us-east-1:ttstrobell</td>
<td>2014-08-1</td>
<td>TRUE</td>
<td>2014-09-2</td>
<td>2014-11-1</td>
<td>FALSE</td>
</tr>
<tr>
<td>woolfc</td>
<td>arn:aws:us-east-1:woolfc</td>
<td>2014-06-1</td>
<td>TRUE</td>
<td>2014-11-0</td>
<td>2014-12-2</td>
<td>FALSE</td>
</tr>
<tr>
<td>ystammi</td>
<td>arn:aws:us-east-1:ystammi</td>
<td>2014-08-1</td>
<td>TRUE</td>
<td>2014-09-2</td>
<td>2014-11-1</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Control resource access

1. **Effect** *(Required)* – specifies whether the statement will explicitly allow *(“Allow”)* or deny *(“Deny”)* access. These are the only two values that are valid in this element.
2. **Action** *(Required)* – describes the type of access that should be allowed or denied.
3. **Resource** *(Required)* - specifies the object or objects that the statement covers.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": 
        [ "s3:PutObject",
          "s3:GetObject",
          "s3:DeleteObject"
        ],
      "Resource": [ 
        "arn:aws:s3:::dev-bucket/*",
        "arn:aws:s3:::qa-bucket/*"
      ]
    }
  ]
}
```
Automation
Automation

Protect against attacks

Protect against bad configuration

Enforce good configuration
Human vs Machine

Reduce human access  \[s/access/error/g\]

Pace of Innovation…meet Pace of Security Automation

Scalable infrastructure needs scalable security
Let’s compare

Human Bob  Auto Bob
Let’s compare

**Human Bob**
Bob needs coffee to do anything

**Auto Bob**
AutoBob doesn’t like Java in the morning
Let's compare

**Human Bob**
Bob needs coffee to do anything
Bob has a hard time finding the keyboard at 3am

**Auto Bob**
AutoBob doesn’t like Java in the morning
AutoBob is backed by serverless infrastructure 24/7/365
Let's compare

**Human Bob**

- Bob needs coffee to do anything
- Bob has a hard time finding the keyboard at 3am
- Sometimes Bob miss a single unicode exploit per 10k logs

**Auto Bob**

- AutoBob doesn’t like Java in the morning
- AutoBob is backed by serverless infrastructure 24/7/365
- ML and EMR backs AutoBob on-demand when he needs to churn large logfiles
Let’s compare

**Human Bob**
- Bob needs coffee to do anything
- Bob has a hard time finding the keyboard at 3am
- Sometimes Bob miss a single unicode exploit per 10k logs
- Bob doesn’t like missing the same unicode exploit every day

**Auto Bob**
- AutoBob doesn’t like Java in the morning
- AutoBob is backed by serverless infrastructure 24/7/365
- ML and EMR backs AutoBob on-demand when he needs to churn large logfiles
- AutoBob already blocked the source and are now going through logs for other systems
Let’s compare

**Human Bob**

Bob needs coffee to do anything  
Bob has a hard time finding the keyboard at 3am  
Sometimes Bob miss a single unicode exploit per 10k logs  
Bob doesn’t like missing the same unicode exploit every day  
Bob dropped his pager in the pool

**Auto Bob**

AutoBob doesn’t like Java in the morning  
AutoBob is backed by serverless infrastructure 24/7/365  
ML and EMR backs AutoBob on-demand when he needs to churn large logfiles  
AutoBob already blocked the source and are now going through logs for other systems  
AutoBob triggers near real-time upon API detection
Automation simplifies component-based security

```python
def sendAlert(isRoot, message):
    # Placeholder for alert function.
    # This could be Amazon SNS, SMS, Email or adding to a ticket tracking system like Jira or Remedy.
    if CHIME_ALERTS is True:
        print("Sending alerts to chime")
        SSM_CLIENT = boto3.client('ssm')
        chimeMessage = {"Content": message}
        req = urllib2.Request(SSM_CLIENT.get_parameter(Name=CHIME_WEBHOOK_PARAMETER, WithDecryption=True)['Parameter']['Value'])
        req.add_header('Content-Type', 'application/json')
        try:
            response = urllib2.urlopen(req, json.dumps(chimeMessage))
        except urllib2.HTTPError as e:
            print("Access to Chime failed with error: " + e)
```

Integrate cloud security in your tools
Cloud Defense Tools

Security automation

!=

Python | Node | Java | <Insert hip name here>
Managed services

Amazon CloudWatch Events
AWS Lambda
Amazon Kinesis Firehose
Amazon Machine Learning
Amazon GuardDuty
Amazon Macie

And so many more…

Managed services are there to offload you
Your BU’s knows this…
Remediation as defense rocks....but

Failure is always an option...just at lightning speed
Test, test, test

GuardRails
Edge cases
Don’t nuke yourself

Framework/Remediation strategy
## The anatomy of security automation

<table>
<thead>
<tr>
<th>Mode</th>
<th>Section</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiate</td>
<td>React</td>
<td>Config Rules / CloudWatch Events / Log Parsing</td>
</tr>
<tr>
<td></td>
<td>Trigger</td>
<td>Lambda</td>
</tr>
<tr>
<td></td>
<td>Learn</td>
<td>Lambda / CloudWatch Logs</td>
</tr>
<tr>
<td></td>
<td>Priority Action</td>
<td>Restart service, delete user, etc.</td>
</tr>
<tr>
<td>Execution</td>
<td>Forensics</td>
<td>Discover: Who/where/when, allowed to execute?</td>
</tr>
<tr>
<td></td>
<td>Countermeasure</td>
<td>Disable access keys, isolate instance, etc.</td>
</tr>
<tr>
<td></td>
<td>Alert</td>
<td>Text/Page, email, ticket system</td>
</tr>
<tr>
<td></td>
<td>Logging</td>
<td>Database, ticket system, encrypt data?</td>
</tr>
</tbody>
</table>
Demo

**Purpose:**
Prevent misconfiguration of Amazon S3 buckets

**Functionality:**
- Automatically remove public access to S3 bucket/objects
- Support whitelisting of public buckets
- Support alert/notification to email, Amazon Chime, Slack
Other examples

CIS AWS Benchmark
Remediate exposed AWS Credentials
Enforce service state (example CloudTrail)
Enforce immutable infrastructure (OS config/services <-> cloud controls)
Track suspected users near real time
Edge traffic analytics
Instance memory capture (example: Margarita Shotgun @ https://threatresponse.cloud/)
OSS and you

Lots of code out there

Review!

Learn!
Other OSS projects

Some of the projects out there:

• ThreatResponse.cloud [https://threatresponse.cloud](https://threatresponse.cloud)
• Cloud Custodian [https://github.com/capitalone/cloud-custodian](https://github.com/capitalone/cloud-custodian)
• Security Monkey [https://github.com/Netflix/security_monkey](https://github.com/Netflix/security_monkey)
• FIDO [https://github.com/Netflix/Fido](https://github.com/Netflix/Fido)
• CloudSploit [https://github.com/cloudsploit](https://github.com/cloudsploit)
• Prowler [https://github.com/Alfresco/prowler](https://github.com/Alfresco/prowler)
• StreamAlert [https://github.com/airbnb/streamalert](https://github.com/airbnb/streamalert)

And many more…

Have a security automation project/repo…let me know!
AWS and OSS

http://github.com/awslabs

http://github.com/awslabs/aws-security-automation

https://github.com/awslabs/aws-security-benchmark
Cloud defense strategy summary

Visibility
What/When/How

Data flow
Enforce desired path (Edge -> Layers -> Data)

Automation
Detect/React/Remediate

Tooling
Find what works for you and scale!
Online resources

https://aws.amazon.com/security/
https://aws.amazon.com/compliance/
https://aws.amazon.com/security/security-resources/
https://aws.amazon.com/quickstart/#security
Remember

Security is a service team, not a blocker

Security is everyone's job

Allow flexibility and freedom

but control the flow and result.
Q&A