The Art of Automation
Creating A Serverless Threat Intel Bot
What Cloud Migration Feels Like….
$ Ronald Eddings

- **Security Architect at Palo Alto Networks**
  - Case Management and Incident Response
  - Security Automation and Orchestration
  - ChatOps

- **Podcast Host of Hacker Valley Studio**
  - Upgrading The Human (Layer 8)
  - Threat Intelligence
  - Security Architecture
### Cloud Security Finding Types

<table>
<thead>
<tr>
<th>AWS</th>
<th>Google Cloud Platform</th>
<th>Azure</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Backdoor</td>
<td>• Web Security Scanner</td>
<td>• IaaS VMs and servers</td>
</tr>
<tr>
<td>• Behavior</td>
<td>• Anomaly Detection</td>
<td>• Native compute</td>
</tr>
<tr>
<td>• CryptoCurrency</td>
<td>• Event Threat Detection</td>
<td>• Data services</td>
</tr>
<tr>
<td>• PenTest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Persistence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Privilege Escalation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Recon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Resource Consumption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Stealth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Trojan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Unauthorized</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Case Study: Threat Intelligence

**GOALS**
- Keep pace with rapidly scaling cloud environment
- Automate everything that humans don’t need to do
- Provide value to other technology departments

**USE CASES**
- EC2 and account compromise
- Phishing Enrichment and response
- CryptoCurrency Mining
Evolution Of Security Architecture

- SecOps
- Security analyst
- Incident responder

Customer apps
- Firewall
- IAM
- Threat Intel

Email

Email

Firewall
EDR
SIEM
Active Directory
DLP
Threat Intel
Evolution Of Security Architecture
Evolution Of Security Architecture

- SecOps
- Security analyst
- Incident responder
- Firewall
- Email
- EDR
- DLP
- SIEM
- Threat Intel
- Customer apps
- IAM
ChatOps
serverless framework

The easy and open way to build serverless applications

aws Azure Google Cloud Platform IBM Cloud kubernetes

serverless inc.
Serverless Operations

- Serverless deploy
- Serverless invoke
- Serverless remove
Getting Started

1. Clone Threat-Intel-Slack-Bot
   ○ git clone https://github.com/secdevopsai/Threat-Intel-Slack-Bot.git
2. Install NodeJS
3. Install Serverless
   ○ sudo npm install -g serverless
4. Install Serverless Python Requirements Plugin
   ○ serverless plugin install -n serverless-python-requirements
5. Configure AWS Account
6. Install Docker
7. Create Slack Workspace
8. Create Slack App
9. Deploy
   ○ serverless deploy
Bot Interaction

 Threat Intel Bot

/threatintel secdevops.ai
Enrich individual or in bulk
providers:
  name: aws
  runtime: python3.7
  region: us-west-1
  stage: dev

environment:
  SLACK_API_TOKEN: "$\{env:SLACK_API_TOKEN\}"
  SLACK_CHANNEL: "$\{env:SLACK_CHANNEL\}"

iamRoleStatements:
  - Effect: "Allow"
    Action:
      - "lambda:InvokeFunction"
    Resource:
      - "*"
      - "arn:aws:lambda:*
        - Ref: AWS::Region
      - Ref: AWS::AccountId
      - "functions:($\{self:service\})-($\{self:provider.stage\})-*"

functions:
  slashmessage:
    handler: lambda_handler.handle_slash_message
    reservedConcurrent: 5
    environment:
      VIRUSTOTAL_LAMBDA_FUNCTION: "$\{self:service\}-($\{self:provider.stage\})-virustotal"
      PASSIVETOTAL_LAMBDA_FUNCTION: "$\{self:service\}-($\{self:provider.stage\})-passivetotal"
    events:
      - http:
          path: slashmessage
          method: any
          requests:
            passThrough: WHEN_NO_MATCH
  passivetotal:
    handler: lambda_handler.handle_passivetotal
    reservedConcurrent: 5
    environment:
      PASSIVETOTAL_USERNAME: "$\{env:PASSIVETOTAL_USERNAME\}"
      PASSIVETOTAL_APIKEY: "$\{env:PASSIVETOTAL_APIKEY\}"
  virustotal:
    handler: lambda_handler.handle_virustotal
    reservedConcurrent: 5
    environment:
      VIRUSTOTAL_API_KEY: "$\{env:VIRUSTOTAL_API_KEY\}"

plugins:
  - serverless-python-requirements

assets:
  PythonRequirements:
    dockerizePip: non-linux
def handle_slash_message(event, context):
    from urllib.parse import parse_qs
    from tisb.slashmessage import handler as slash_handler

    slash_message = {
        k: v[0] for k, v in parse_qs(event['body']).items()
    }
    channel = slash_message['channel_id']
    indicator = slash_message.get('text', '')
    return slash_handler(indicator=indicator, channel=channel)

def handle_passivetotal(event, context):
    from tisb.passivetotal import handler as passivetotal_handler
    return passivetotal_handler(**event)

def handle_virustotal(event, context):
    from tisb.virustotal import handler as virustotal_handler
    return virustotal_handler(**event)
source: ['pingly', 'riskiq']

lastSeen: 2019-04-05 12:38:24

resolveType: domain

firstSeen: 2015-05-12 10:52:10

collected: 2019-06-28 17:20:06

recordType: NS

recordHash: 1933fa4a44f4e692617953ab6492d679200ed0132ed1a440ef596a03939b3a

resolve: ns-cloud-c4.googledomains.com

value: demisto.com

13 replies Last reply 3 months ago
PassiveTotal Results for secdops.ai
Query Results for 52.4.240.221
Resolve
52.4.240.221
Resolve Type
ip
Record Type
A
First Seen
2017-11-12 09:50:33
Last Seen
2019-11-03 18:10:18

PassiveTotal
Query Results for alt2.aspmx.l.google.com
Resolve
alt2[][aspmx][.]l[.]google[.]com
Resolve Type
domain
Record Type
MX
First Seen
2018-03-04 14:20:17
Last Seen
2019-11-03 18:10:18
EC2 Instance is communicating with a disallowed IP Address

Unauthorized Access - EC2 Malicious IP

Guard Duty Findings

Severity
Medium

Region
us-west-2

Count
13

IP Address
54.70.213.192
New Guard Duty Finding:

EC2 Instance is communicating with a disallowed IP Address

Severity: Medium
Count: 13
Region: us-west-2
IP Address: 54.70.213.192

[Buttons: Approve, Disable Public IP Address]
Bot Security

- Event injection
- Broken authentication
- Insecure deployment settings
- Misuse of permissions and roles
- Insufficient logging
- Insecure storing of app secrets
- DoS attacks and financial exhaustion
- Improper exception handling
services: threat-intel-bot

providers:
  name: aws
  runtime: python3.7
  region: us-west-1
  stage: dev

environment:
  SLACK_API_TOKEN: "${env:SLACK_API_TOKEN}"
  SLACK_CHANNEL: "$[env:SLACK_CHANNEL]"

iamRoleStatements:
  - Effect: "Allow"
    Action:
      - "lambda:InvokeFunction"
    Resource:
      - Fn::Join:
          - "-"
          - - "arn:aws:lambda"
          - Ref: "AWS::Region"
          - Ref: "AWS::AccountId"
          - "functions:${self:service}-${self:provider.stage}-"
Where To Go Next

- Enhance SecOps and DevOps workflows with serverless architecture
- Implement cloud functions in Playbooks / Step Functions
- SecOps and DevOps self service via Chat Bot Dialogs
- Curate data by analysts and engineers for Machine Learning
Conclusion