nccgroup

AWS Security

Staying on Top of the Cloud

Intro

- **Kurtis Miller**, a Principal Security Consultant for nccgroup
- Previously:
  - Global Security Engineering Manager for Atlassian
  - Senior Security Consultant for iSECpartners part of nccgroup
  - First Security Engineer for HomeAway
  - An IT Manager, IT Consultant, and security hobbyist.
- I am **noperand** on Twitter, GitHub, Freenode, and elsewhere.

Agenda

- **Some AWS Fundamentals**
- **Tools, Code, and a Demo**
- **A Tale About Alice and Bob**

Amazon Web Services (AWS)

- **AWS or Amazon Web Services** is a group of several different cloud-based services designed for solving a variety of problems.
  - **EC2** or **Elastic Compute Cloud**
  - **IAM** or **Identity and Access Management**
  - **S3** or **Simple Storage Service**
- There are many other AWS services, much more functionality and much, much more complexity, but this simple list covers enough common fails.

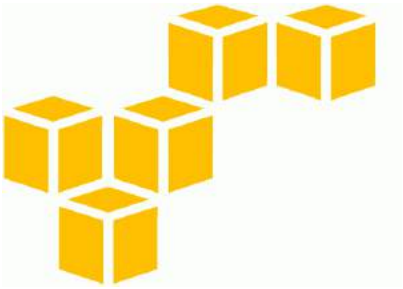
Amazon Elastic Compute Cloud (EC2)

- Provides virtual environments to host your applications.
- Provides a slew of templates or Amazon Machine Images (AMIs) you can choose from or build and your own.
- Flexible storage options.
- Can be public or private, leveraging public IP addresses or a Virtual Private Cloud (VPC).
- Firewall support through Security Groups.

Amazon Simple Storage Service (S3)

- Create buckets to organize your data.
- Throw objects (files) into the buckets.
- Assign permissions to buckets and/or individual objects (can be tricky).
- All available through a RESTful API.
- Supports features like versioning and Reduced Redundancy Storage (RRS).

Amazon Identity and Access Management (IAM)



- Centrally create users and issue key pairs for API-level access.
- Configure policy and access control for all AWS services.
- Enable Multi-Factor Authentication (MFA).
- Offers an API, just like all other services.
- Also offers an interactive console.

Approach for … Most?

- ***"Move fast and break things."***
- Developers get access and are let loose to ***"innovate"!***
  - This often includes agitating the security team, if you have one.
- Secrets and credential management? Relatively nil if done at all.
- In many cases, it comes down to "*What's the bill this month?*" and proceeds from there.
- Many organizations centralize their cloud infrastructure in AWS to consolidate billing (and risk).
  - There be dragons.

There has to be a better way!

- Utopian case? Observe best practice, establish some kind of policy and enforce it.
- Leverage tools/automation to tell you when things aren't right and deal with it.
- Assess on a regular schedule to identify security relevant changes.
- Visualize what is going on for quicker response.
- Improve overall situational awareness around use of AWS.

Tools

- truffleHog
  - https://github.com/dxa4481/truffleHog
- AWS-Recipes
  - https://github.com/nccgroup/AWS-recipes
- Scout2
  - https://github.com/nccgroup/Scout2
- ***...and custom code!***

truffleHog

nccgroup

- Goal is to find high-entropy strings that might be used for authentication or crypto.
- Helps to locate things that developers should not be putting into version control.
- Goes through Git repositories including all commits and branches to find high entropy strings.
- Great for finding AWS secret keys and finding some other interesting strings.
- Enables covering a lot of ground very quickly.
- Mostly false positives, unfortunately.
- Absolutely requires a human to operate, every time.
- Your mileage may vary.

truffleHog Examples

```
-<tag name="patch-summary-cves-72d764fe23f20f68ad54e7ad09ec98cf">CVE-2016-6515, CVE-2016-6210, CVE
-2016-3115, CVE-2015-5600, CVE-2015-5352, CVE-2014-2653</tag>
-<tag name="patch-summary-cve-num-72d764fe23f20f68ad54e7ad09ec98cf">6</tag>
-<tag name="patch-summary-txt-72d764fe23f20f68ad54e7ad09ec98cf">OpenSSH &lt; 7.4 Multiple Vulnerab
```

```
tr><td><table xmlns="" cellpadding="0" cellspacing="0" border="0" width="100%">
-<tr><td class="nopadding"><h1 class="classtitle"><img src="
AAAAAP///z1QXwCktff3+07v8cfM0d7h5N3g49zf4trd40rs7ujq70fp6+Tm6Pf4+fX29/T19vP09fDx8pGbpJ2mrkFTYkNVZE
tcak1ebE9gblJjcVVlclhodVpqd1lpdl5ue2h2gmt5hW17h3aDjnWCjXSBjHmGkXeEj4GNl4WRm4uWn4qVnoiTnJGcpY+ao46Z
```

```
-pass: JejiZrEe(kL9ff[redacted]ZVy3uy*fu3%bsVrn9ex
-
-Access key ID          Secret access key
-A[redacted]A   mD8fiX2Ax[redacted]5NjPC8js
```

Didn't flag password

FLAGGED SECRET KEY!!!

Didn't flag Access Key

AWS-Recipes

nccgroup

- Provides a number of IAM policies and Python-based tools to help with management and maintenance of an AWS environment.
- Automates and abstracts user deletion, MFA enablement, and key rotation, among other things.
- Single dependency on **opinel**, a core Python module that is also used by Scout2.

AWS-Recipes

nccgroup

aws_cloudtrail_enable_all_regions....	Complete fix for issue #5	5 months ago
aws_cloudtrail_get_logs.py	More fixes for p3	3 months ago
aws_ec2_empty_default_security_g..	p3-style exceptions	3 months ago
aws_iam_create_default_groups.py	Complete fix for issue #5	5 months ago
aws_iam_create_policy.py	Complete fix for issue #5	5 months ago
aws_iam_create_user.py	Complete fix for issue #5	5 months ago
aws_iam_delete_user.py	Complete fix for issue #5	5 months ago
aws_iam_enable_mfa.py	Fixes to iam_enable_mfa and configure_iam	4 months ago
aws_iam_rotate_my_key.py	Update rotate_my_key, should work now	3 months ago
aws_iam_sort_users.py	Complete fix for issue #5	5 months ago
aws_recipes_assume_role.py	Complete fix for issue #5	5 months ago
aws_recipes_configure_iam.py	Do not automatically rename the profile when saving the MFA serial nu...	3 months ago
aws_recipes_create_ip_ranges.py	Fix arg name	3 months ago
aws_recipes_get_permissions.py	Complete fix for issue #5	5 months ago
aws_recipes_init_sts_session.py	Complete fix for issue #5	5 months ago

Scout2

- Configure a read-only IAM profile. This can be found in the AWS-Recipies GitHub repo.
  - https://github.com/nccgroup/AWS-recipes/blob/master/IAM-Policies/Scout2-Default.json
- Pulls configuration of all AWS services and evaluates the configuration using rules and a rule engine implemented Python.
- Produces JavaScript that assigns the results to a JSON object.
- Front-end code represents these results in some colorful ways.

Scout2

nccgroup

Scout2 | CloudTrail | EC2 | IAM | RDS | Redshift | S3 | SNS | Regions | Filters | Help

EC2 Dashboard

Default security groups in use
- Security groups checked: 178
- Security groups flagged: 4

Non-empty rulesets for default security groups
- Rulesets checked: 356
- Rulesets flagged: 32

DNS port open to all
- Rules checked: 1568
- Rules flagged: 2

MongoDB port open to all
- Rules checked: 1568
- Rules flagged: 0

MsSQL port open to all
- Rules checked: 1568
- Rules flagged: 0

MySQL port open to all
- Rules checked: 1568
- Rules flagged: 0

NFS port open to all
- Rules checked: 1568
- Rules flagged: 0

Oracle DB port open to all
- Rules checked: 1568
- Rules flagged: 0

PostgreSQL port open to all
- Rules checked: 1568
- Rules flagged: 0

RDP port open to all
- Rules checked: 1568
- Rules flagged: 0

SMTP port open to all
- Rules checked: 1568
- Rules flagged: 0

SSH port open to all
- Rules checked: 1568
- Rules flagged: 7

TCP port open to all
- Rules checked: 1568
- Rules flagged: 53

UDP port open to all
- Rules checked: 1568
- Rules flagged: 2

All ports open
- Rules checked: 1216
- Rules flagged: 21

All ports open to all

Unrestricted network traffic within security group

FTP port open

Scout2

nccgroup

Scout2 CloudTrail EC2 IAM RDS Redshift S3 SNS Regions Filters Help

IAM Dashboard

Cross-account AssumeRole policy lacks external ID
- Roles checked: 68
- Roles flagged: 0

AssumeRole policy allows all principals
- Roles checked: 68
- Roles flagged: 0

Unused role for EC2
- Roles checked: 0
- Roles flagged: 0

Group with inline policies
- groups checked: 24
- groups flagged: 16

Inline group policy allows NotActions
- Policies checked: 83
- Policies flagged: 1

Inline group policy allows iam:PassRole *
- Policies checked: 83
- Policies flagged: 0

Inline group policy allows sts:AssumeRole *
- Policies checked: 83
- Policies flagged: 1

Inline role policy allows NotActions
- Policies checked: 29
- Policies flagged: 0

Inline role policy allows iam:PassRole *
- Policies checked: 29
- Policies flagged: 0

Inline role policy allows sts:AssumeRole *
- Policies checked: 29
- Policies flagged: 0

Inline user policy allows NotActions
- Policies checked: 729
- Policies flagged: 0

Inline user policy allows iam:PassRole *
- Policies checked: 729
- Policies flagged: 0

Inline user policy allows sts:AssumeRole *
- Policies checked: 729
- Policies flagged: 0

Managed policy allows NotActions
- Policies checked: 142
- Policies flagged: 0

Managed policy allows iam:PassRole *
- Policies checked: 142
- Policies flagged: 9

Managed policy allows sts:AssumeRole *

Minimum password length too short

Password expiration disabled

What We Often See

- Lack of Multi-Factor Authentication (MFA).
- Security Groups allowing all inbound traffic to EC2 instances.
- IAM policies allowing free assumption and passing of roles facilitating elevation of privilege.
- Lack of access key rotation.
- Poor password policies configured in IAM.
- S3 buckets that are accessible by all authenticated AWS users (even outside your organization).

Expanding Upon Scout2

nccgroup

- Scout2 gathers up your AWS configuration information and provides an attack surface report.
- This is represented by the EC2 instances that you currently have deployed and your configured Security Groups.
- What if we produced host and service discovery shell scripts based on this information? It would certainly prevent us from scanning things we shouldn’t, from an IP address and port standpoint.

Expanding Upon Scout2

```
f = open(AWSCONFIG, "r")
l = f.readline() # HACK: skip first line assignment

data = json.load(f)
attack_surface = data['services']['ec2']['attack_surface']

# why would this be anything but a bash script?
print "#!/bin/bash"

# iterate per IP address and build lists of TCP and UDP ports
# allowed according to Security Groups
for ip in attack_surface:
    tcp_ports = []
    udp_ports = []

    entity = attack_surface[ip]

    # the port numbers are keys at a specific level in the JSON object
    try:
        # TCP first
        for port in entity['protocols']['TCP']['ports'].keys():
            tcp_ports.append(port)
    except KeyError:
        pass

    try:
        # UDP next
        for port in entity['protocols']['UDP']['ports'].keys():
            udp_ports.append(port)
    except KeyError:
        pass

    # output a constructed nmap command based on the IPs and ports
    if len(tcp_ports) > 0:
        print "sudo nmap -vvv -sV -A -O -Pn -T4 -p " + ",".join(tcp_ports) + ' -oA ' + ip + '.tcp ' + ip
    if len(udp_ports) > 0:
        print "sudo nmap -vvv -sU -A -Pn -T4 -p " + ",".join(udp_ports) + ' -oA ' + ip + '.udp ' + ip

sys.exit(0)
```

Expanding Upon Scout2

```
#!/bin/bash
sudo nmap -vvv -sV -A -O -Pn -T4 -p 443,8080,445,10050 -oA [redacted]tcp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 137-139,22,443,445,0-65535,80 -oA [redacted].82.tcp [redacted]
sudo nmap -vvv -sU -A -Pn -T4 -p 123,137-139 -oA [redacted].udp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 22 -oA [redacted].tcp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 10000-10003,8080,443,5985-5986,445,18080,135,28080,80,10050-[redacted]
sudo nmap -vvv -sU -A -Pn -T4 -p 138,53 -oA [redacted].udp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 5985-5986,3389 -oA [redacted].tcp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 80 -oA [redacted].tcp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 25,22,443,43,465,80 -oA [redacted].tcp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 990,22,40000-40003 -oA [redacted].tcp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 51,50,22 -oA [redacted]4.tcp [redacted]
sudo nmap -vvv -sU -A -Pn -T4 -p 51,4500,500,50 -oA [redacted].udp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 22,8443 -oA [redacted]tcp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 443,8080,445,10050 -oA [redacted].tcp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 5985-5986,3389 -oA [redacted].tcp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 137-139,22,443,445,0-65535,80 -oA [redacted].tcp [redacted]
sudo nmap -vvv -sU -A -Pn -T4 -p 123,137-139 -oA [redacted].udp [redacted]
sudo nmap -vvv -sV -A -O -Pn -T4 -p 443,2598,80,1494,3389 -oA [redacted].tcp [redacted]
```

Expanding Upon Scout2

nccgroup

- Scout2 only represents data from a single AWS account.
- What if you have multiple AWS contexts or products hosted in Scout2?
- What methods are available to represent additional contexts, in the same view, to quickly identify areas that need attention?

- What about carving up the data from Scout2, merging it and using D3.js to visually represent it?

Expanding Upon Scout2

nccgroup

DEMO

Yeah! Everything is Great!

nccgroup

- You frequently assess and monitor your AWS configurations.
- You enforce corporate policies within AWS and hold resource owners accountable.
- You regularly pentest the applications you have exposed in AWS.
- You regularly hunt for secrets being persisted in version control.

Yeah! Everything is Great!

nccgroup

All of this is still not enough.

It’s time for a story.

An AWS Horror Story

nccgroup

- Some suspicious activity was identified in logs.
- We investigate, establish confirmed IOCs, burn all creds in assessed scope of compromise, and rotate access keys.
- Patient zero was derived simply from our timeline (being first) with no strong links to the others that were compromised.
  - Rip apart laptops for malware, interview multiple users, etc.
- We start incrementally creeping back to BAU.
- Continue the hunt for malicious activity with available IOCs.

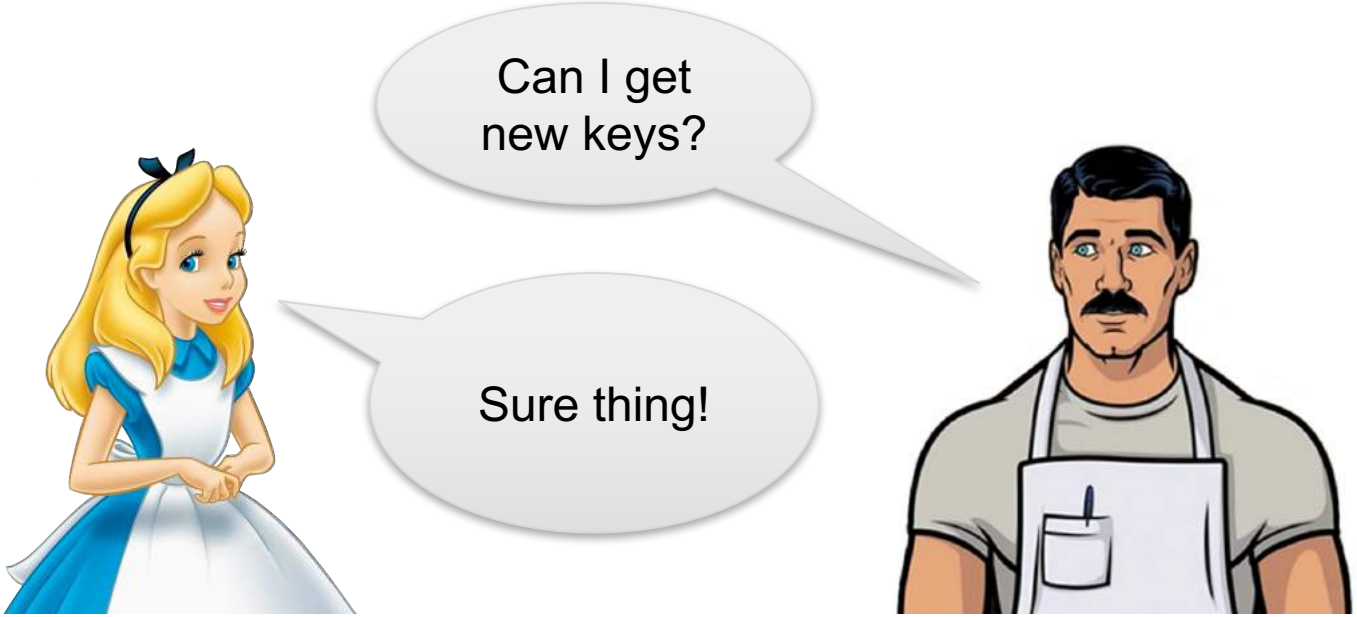
All was calm, until...

An AWS Horror Story

- More hits on some glaring IOCs, after scorched creds.
- The attacker is taunting us. We haven't found their way in yet.
- After several more hours of staring at data and a whiteboard, we trace events and creds to a common thread.

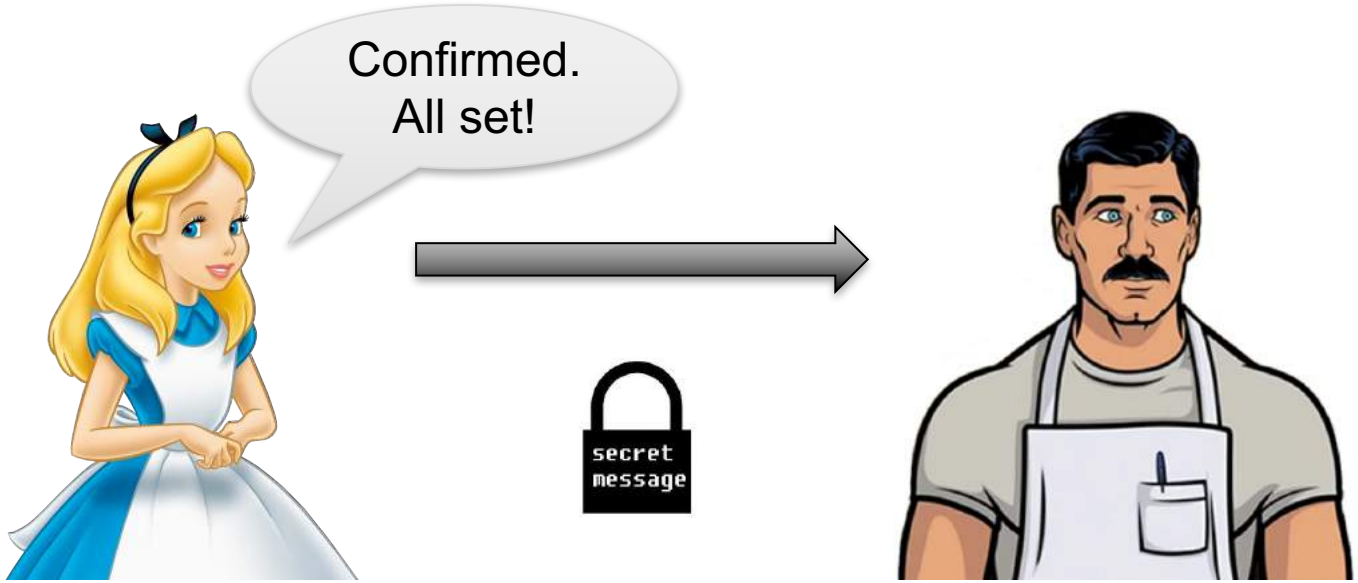
The Tale of Alice and Bob

- **Bob's keys got burned as part of our response.**
- **Bob needs to get new keys from Alice.**
- **Alice generates a new key pair for Bob.**



The Tale of Alice and Bob

- **Alice sends the new keys to Bob as a Secure Message, using Solution.**
- **According to Solution, only Alice and Bob have permission to view the Secure Message.**
- **Alice views the Secure Message once, after submission.**



The Tale of Alice and Bob

- **Bob never views the Secure Message.**
- **Eve uses Bob's new keys.**

Eve is Sly

- Compromised endpoints/users? No signs of malware or other access.
- Various users, no indicators of an insider making moves.
- Majority of compromised credentials were shared the same way, through Solution.

Bob's Your Uncle!!! No, wait...

- Turns out, Maker was logging all API interactions in Solution, including the content of Secure Messages.
- Third-party logging provider used by Maker made their logs accessible directly over the Internet through a nice web UI.
- Third-party logging web interface account was compromised through a common password (a la password DB leaks; think LinkedIn, RockYou, etc).
- All initial points of compromise could be traced to Secret Messages in Maker Solution.

Bob's Your Uncle!!! No, wait...

nccgroup

The initial (and persistent) compromise was completely outside of our environment and thus,

OUTSIDE OF OUR CONTROL.

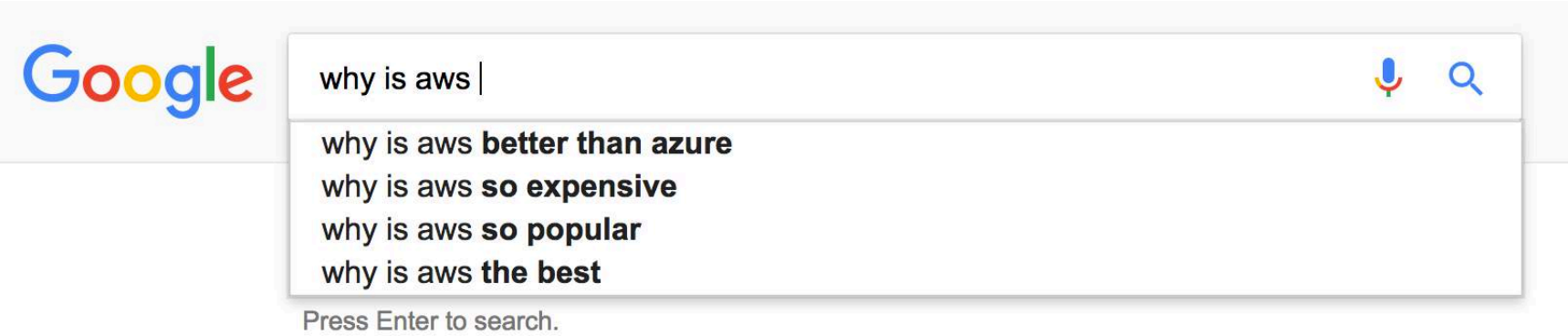
We still managed to work with Maker to improve Solution and get a good handle on the situation.

Ugh, Is There a Moral Here?!

- **While you strive to do right you will be wronged.**
- Stay on top of your infrastructure, holding vendors and third-parties accountable.
- Go through the exercise of asking for detailed logs from vendors.
    - Especially for communication systems your users trust.
- What you may, or may not find, might astound you.

- Leverage the tools and techniques we've described to stay on top of how AWS is being used in your environment.
- Extend your corporate policies to AWS usage and enforce those policies.

Questions?


Google
why is aws
why is aws better than azure
why is aws so expensive
why is aws so popular
why is aws the best
Press Enter to search.



North America

Atlanta

Austin

Chicago

New York

San Francisco

Seattle

Sunnyvale

Europe

Manchester - Head Office

Amsterdam

Cheltenham

Copenhagen

Edinburgh

Glasgow

Leatherhead

London

Luxembourg

Milton Keynes

Munich

Zurich

Australia

Sydney