GitLab CTF Beginner Challenges Workshop /w DC604 and Shyhat

This workshop session will cover a number of easier CTF challenges from the GitLab CTF event from March 2020. More specifically, the names of the challenges involved and the type of exploit involved in solving them is the following:

- Sea-Surf 1-3 (Server-side request forgery)
- Tar2zip (Linux file symbolic links)
- GTP & OTP (source-code review)

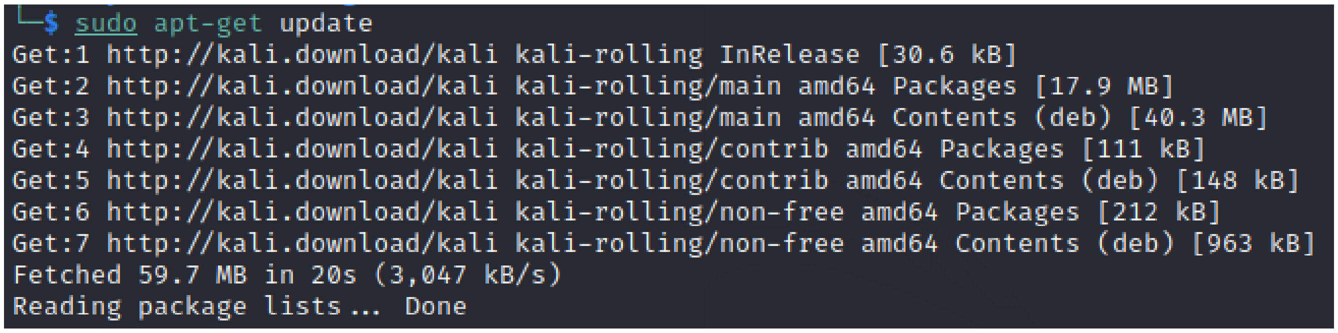
Pre-requisites / Setup

This workshop assumes that participants have access to an up-to-date version of Kali Linux, either as a "bare metal" host OS or a Virtual Machine. Installation of Docker and Docker-Compose is also required, so we will walk through the installation process below.

Part 1 – Installing Docker

Step 1 – update the apt package manager:

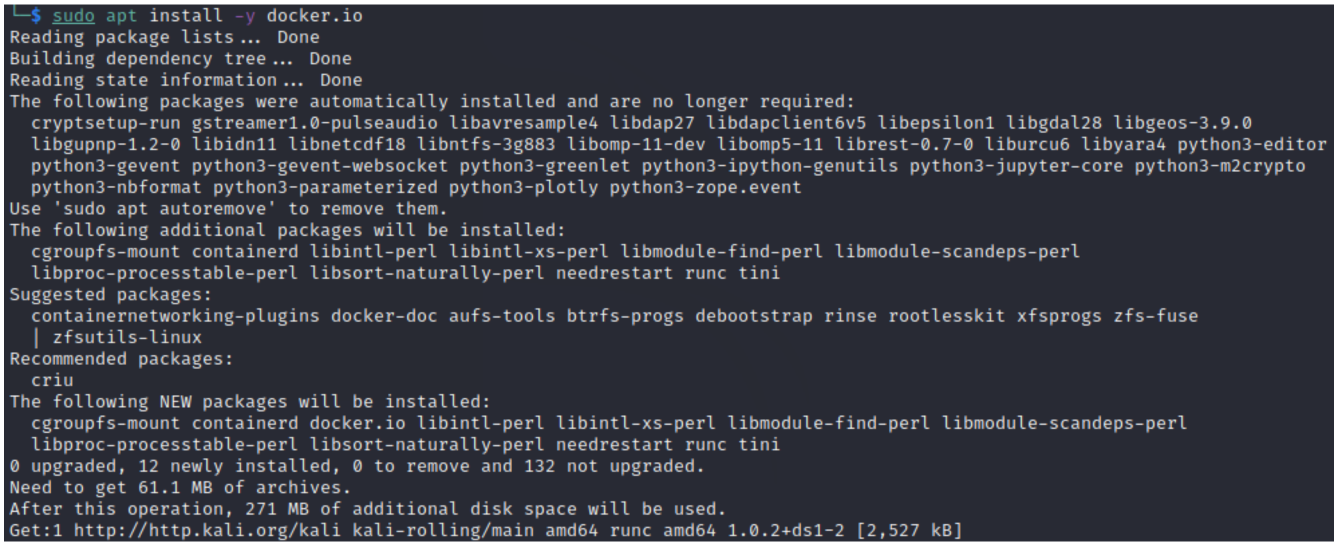
sudo apt update



```
└─$ sudo apt-get update
Get:1 http://kali.download/kali kali-rolling InRelease [30.6 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [17.9 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [40.3 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [111 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [148 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [212 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [963 kB]
Fetched 59.7 MB in 20s (3,047 kB/s)
Reading package lists ... Done
```

Step 2 – install the Docker package:

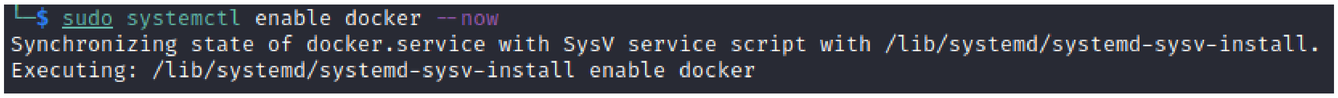
sudo apt install -y docker.io



```
└─$ sudo apt install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  cryptsetup-run gstreamer1.0-pulseaudio libavresample4 libdap27 libdapclient6v5 libepsilon1 libgdal28 libgeos-3.9.0
  libgupnp-1.2-0 libidn11 libnetcdf18 libntfs-3g883 libomp-11-dev libomp5-11 librest-0.7-0 liburcu6 libyara4 python3-editor
  python3-gevent python3-gevent-websocket python3-greenlet python3-ipython-genutils python3-jupyter-core python3-m2crypto
  python3-nbformat python3-parameterized python3-plotly python3-zope.event
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  cgroupfs-mount containerd libintl-perl libintl-xs-perl libmodule-find-perl libmodule-scandeps-perl
  libproc-processtable-perl libsort-naturally-perl needrestart runc tini
Suggested packages:
  containernetworking-plugins docker-doc aufs-tools btrfs-progs debootstrap rinse rootlesskit xfsprogs zfs-fuse
  | zfsutils-linux
Recommended packages:
  criu
The following NEW packages will be installed:
  cgroupfs-mount containerd docker.io libintl-perl libintl-xs-perl libmodule-find-perl libmodule-scandeps-perl
  libproc-processtable-perl libsort-naturally-perl needrestart runc tini
0 upgraded, 12 newly installed, 0 to remove and 132 not upgraded.
Need to get 61.1 MB of archives.
After this operation, 271 MB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 runc amd64 1.0.2+ds1-2 [2,527 kB]
```

Step 3 – Enable the Docker service:

sudo systemctl enable docker --now

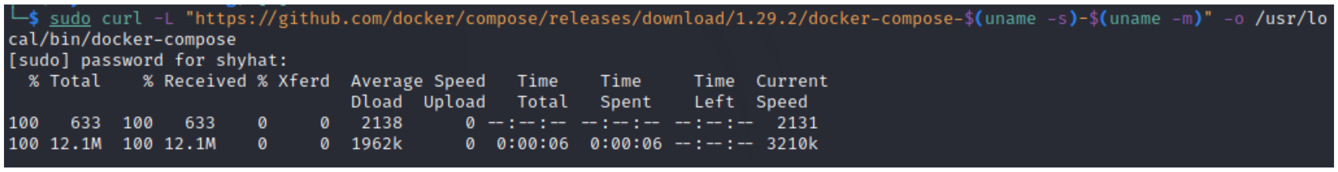


```
└─$ sudo systemctl enable docker --now
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
```

Part 2 – Installing Docker-Compose

Step 1 – Download the Docker-Compose binary with cURL:

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose



```
└─$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/lo
cal/bin/docker-compose
[sudo] password for shyhat:
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   633  100   633    0     0   2138      0 --:--:-- --:--:-- --:--:--  2131
100 12.1M  100 12.1M    0     0  1962k      0  0:00:06  0:00:06 --:--:-- 3210k
```

Step 2 – Enable the Docker-Compose binary's executable permissions:

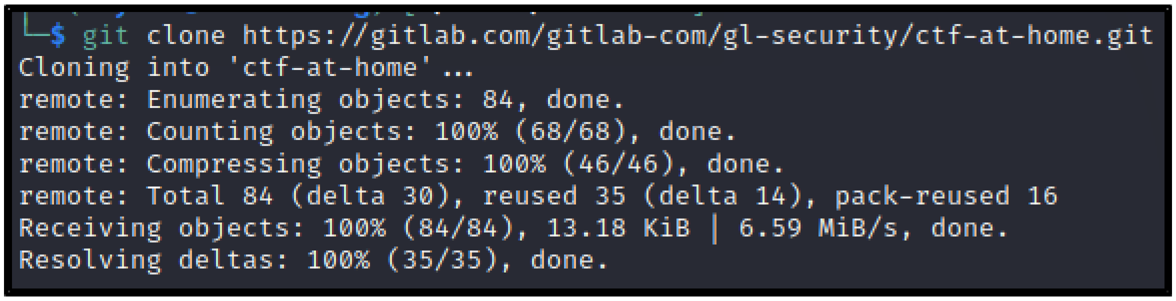
sudo chmod +x /usr/local/bin/docker-compose

NOTE: The command to install Docker-Compose is specific to version 1.29.2. If you have problems installing Docker-Compose, instructions for installation can be obtained from the following URL: https://docs.docker.com/compose/install/

Part 3 – Setup the GitLab CTF Docker-Compose Repository

Step 1 - the following command downloads the GitLab CTF's Git repository:

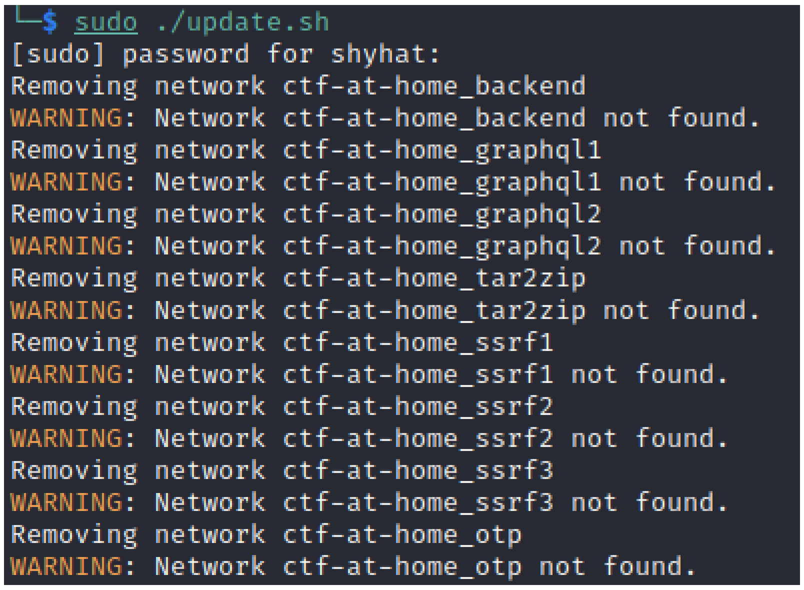
git clone https://gitlab.com/gitlab-com/gl-security/ctf-at-home.git



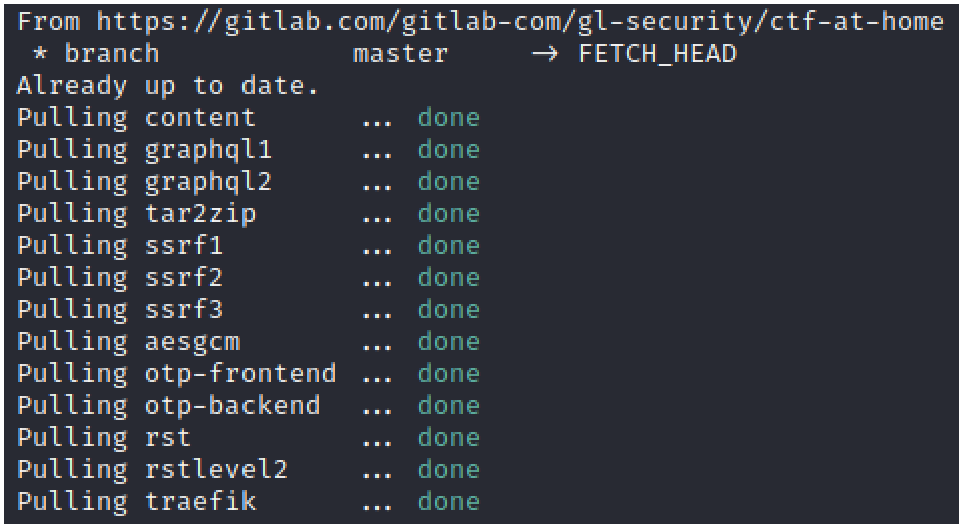
```
└─$ git clone https://gitlab.com/gitlab-com/gl-security/ctf-at-home.git
Cloning into 'ctf-at-home'...
remote: Enumerating objects: 84, done.
remote: Counting objects: 100% (68/68), done.
remote: Compressing objects: 100% (46/46), done.
remote: Total 84 (delta 30), reused 35 (delta 14), pack-reused 16
Receiving objects: 100% (84/84), 13.18 KiB | 6.59 MiB/s, done.
Resolving deltas: 100% (35/35), done.
```

Step 2 – change directory to the ctf-at-home directory and then run the update.sh script:

cd ctf-at-home
sudo ./update.sh



```
└─$ sudo ./update.sh
[sudo] password for shyhat:
Removing network ctf-at-home_backend
WARNING: Network ctf-at-home_backend not found.
Removing network ctf-at-home_graphql1
WARNING: Network ctf-at-home_graphql1 not found.
Removing network ctf-at-home_graphql2
WARNING: Network ctf-at-home_graphql2 not found.
Removing network ctf-at-home_tar2zip
WARNING: Network ctf-at-home_tar2zip not found.
Removing network ctf-at-home_ssrf1
WARNING: Network ctf-at-home_ssrf1 not found.
Removing network ctf-at-home_ssrf2
WARNING: Network ctf-at-home_ssrf2 not found.
Removing network ctf-at-home_ssrf3
WARNING: Network ctf-at-home_ssrf3 not found.
Removing network ctf-at-home_otp
WARNING: Network ctf-at-home_otp not found.
```

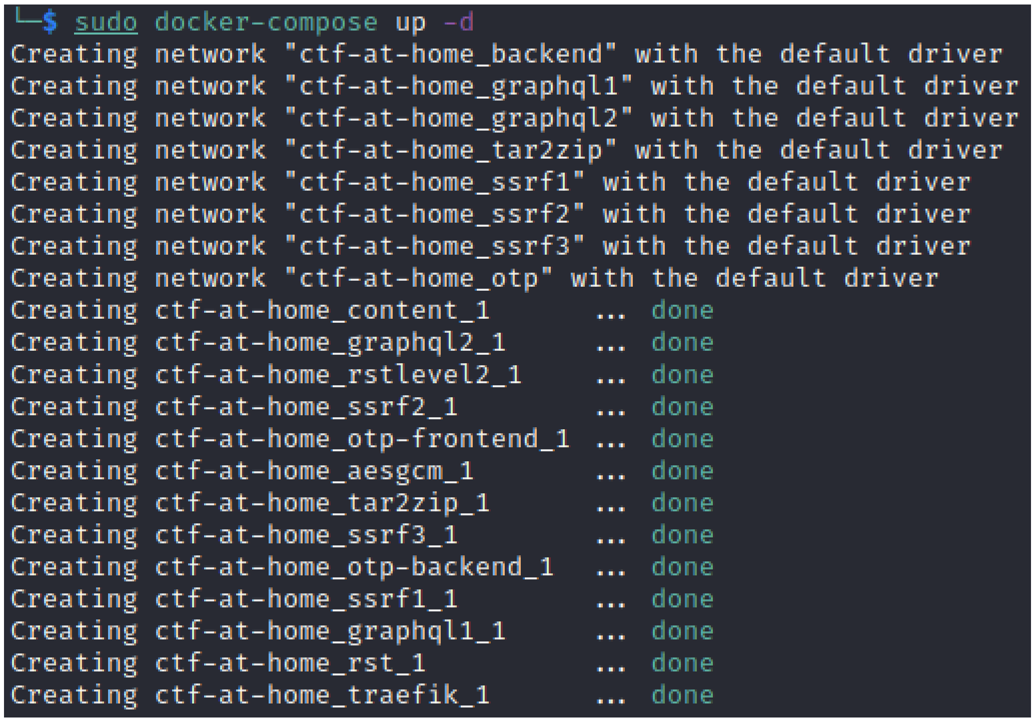


```
From https://gitlab.com/gitlab-com/gl-security/ctf-at-home
 * branch            master     -> FETCH_HEAD
Already up to date.
Pulling content      ... done
Pulling graphql1     ... done
Pulling graphql2     ... done
Pulling tar2zip      ... done
Pulling ssrf1        ... done
Pulling ssrf2        ... done
Pulling ssrf3        ... done
Pulling aesgcm       ... done
Pulling otp-frontend ... done
Pulling otp-backend  ... done
Pulling rst          ... done
Pulling rstlevel2    ... done
Pulling traefik      ... done
```

Part 4 – Start the Docker-Compose Containers

Step 1 – use the following command to start the containers:

sudo docker-compose up -d

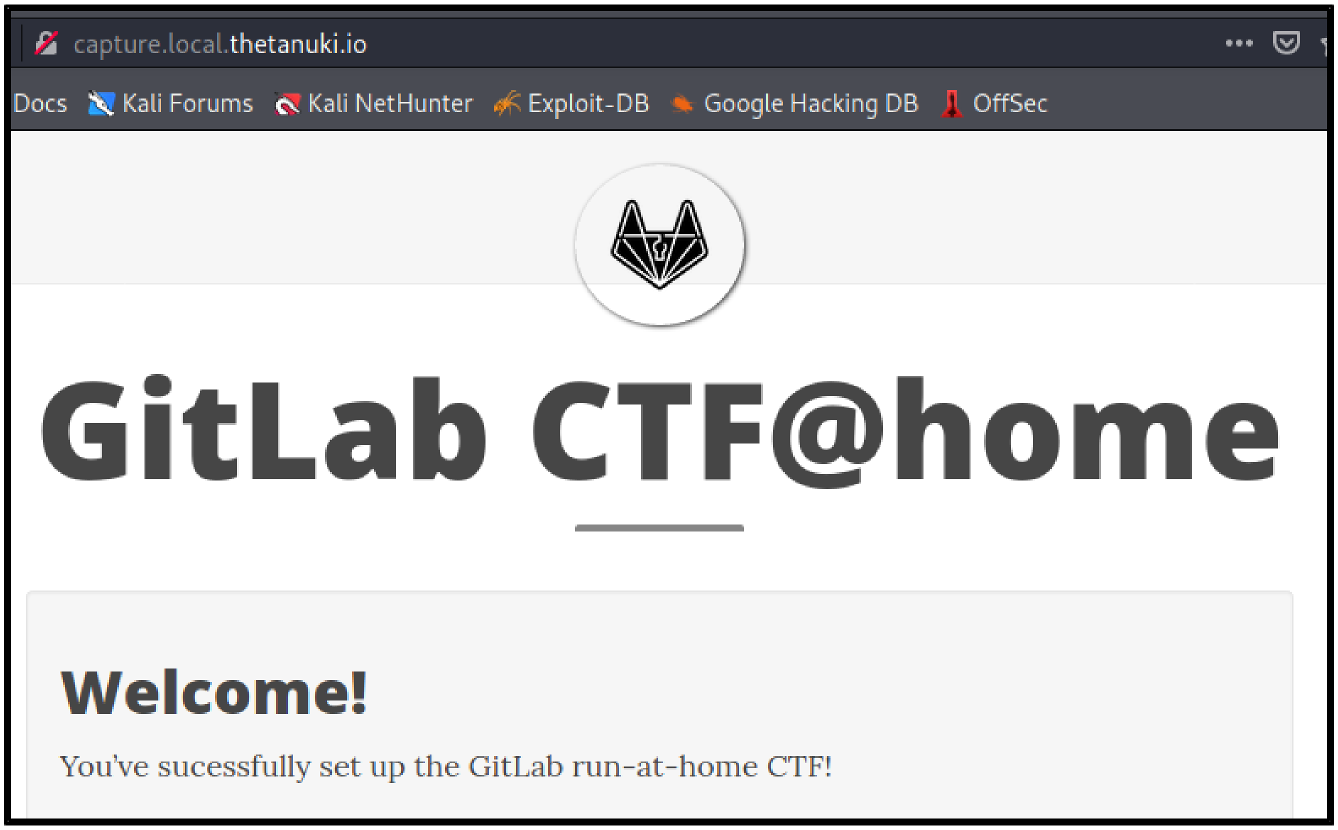


```
└─$ sudo docker-compose up -d
Creating network "ctf-at-home_backend" with the default driver
Creating network "ctf-at-home_graphql1" with the default driver
Creating network "ctf-at-home_graphql2" with the default driver
Creating network "ctf-at-home_tar2zip" with the default driver
Creating network "ctf-at-home_ssrf1" with the default driver
Creating network "ctf-at-home_ssrf2" with the default driver
Creating network "ctf-at-home_ssrf3" with the default driver
Creating network "ctf-at-home_otp" with the default driver
Creating ctf-at-home_content_1      ... done
Creating ctf-at-home_graphql2_1     ... done
Creating ctf-at-home_rstlevel2_1    ... done
Creating ctf-at-home_ssrf2_1        ... done
Creating ctf-at-home_otp-frontend_1 ... done
Creating ctf-at-home_aesgcm_1       ... done
Creating ctf-at-home_tar2zip_1      ... done
Creating ctf-at-home_ssrf3_1        ... done
Creating ctf-at-home_otp-backend_1  ... done
Creating ctf-at-home_ssrf1_1        ... done
Creating ctf-at-home_graphql1_1     ... done
Creating ctf-at-home_rst_1          ... done
Creating ctf-at-home_traefik_1      ... done
```

Part 5 – Access the GitLab CTF Webpage

Step 1 – open a web browser (e.g. Firefox) and navigate to the following URL:

http://capture.local.thetanuki.io



GitLab CTF@home

Welcome!

You've sucessfully set up the GitLab run-at-home CTF!

Sea-Surf 1

Objective:
Have the application communicate with itself

Key Information:
The application takes input in the form of an absolute URL.

Part 6 – Navigate to the Sea-Surf 1 page:

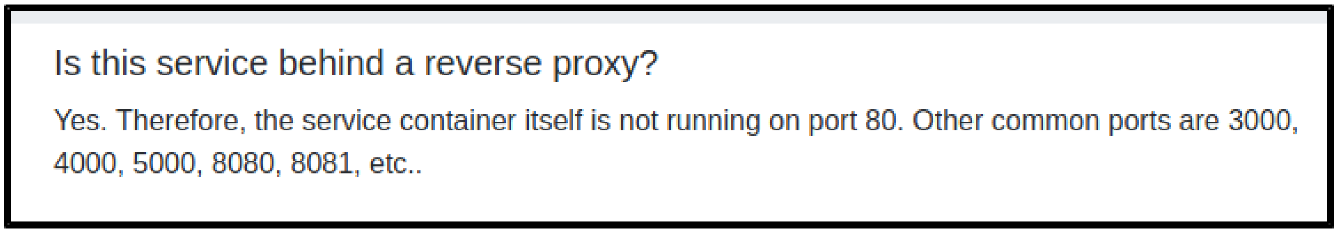
Step 1 – Go to the following page:

http://ssrf1.local.thetanuki.io/

Part 7 – Check Out the FAQ Page

Step 1 – Go to the following page:

http://ssrf1.local.thetanuki.io/faq



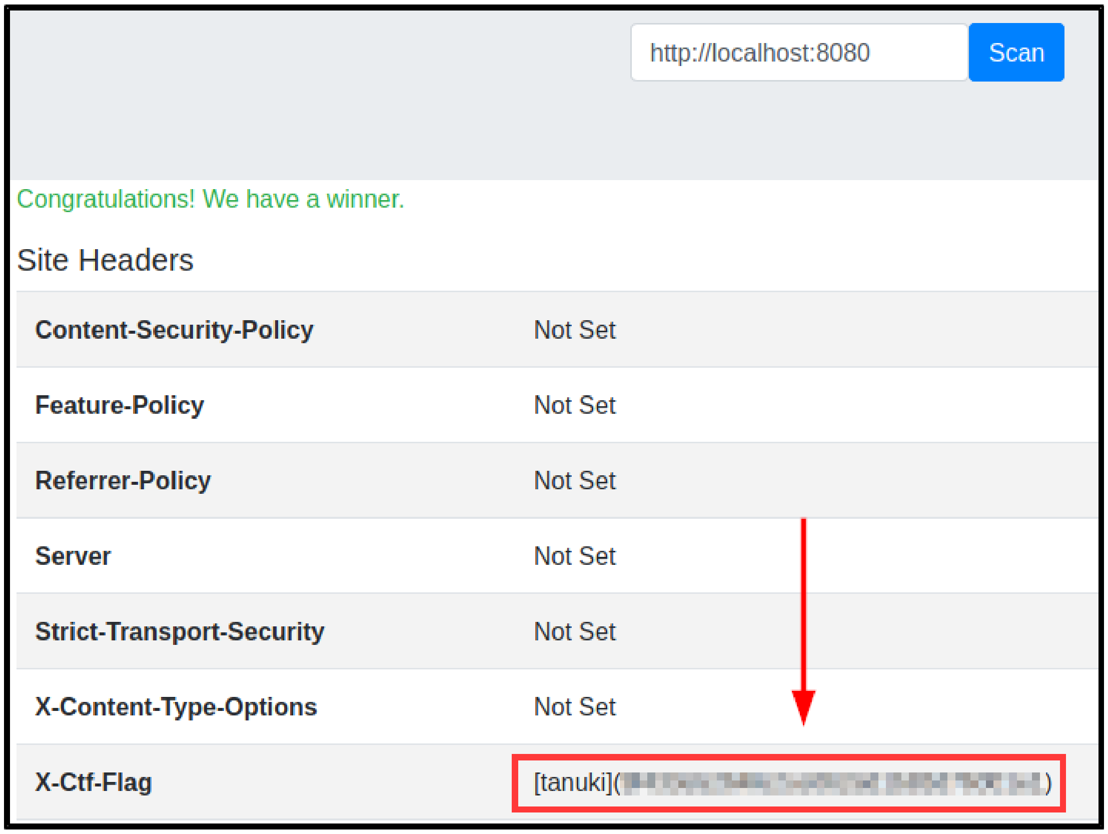
Is this service behind a reverse proxy?

Yes. Therefore, the service container itself is not running on port 80. Other common ports are 3000, 4000, 5000, 8080, 8081, etc..

Part 8 – Execute a Server-Side Request Forgery Attack

Step 1 – Input the following into the **Scan a site** field and click **Scan**:

http://localhost:8080



Congratulations! We have a winner.

Site Headers

Content-Security-Policy	Not Set
Feature-Policy	Not Set
Referrer-Policy	Not Set
Server	Not Set
Strict-Transport-Security	Not Set
X-Content-Type-Options	Not Set
X-Ctf-Flag	[tanuki]([redacted])

Context:

A server-side request forgery (SSRF) attack occurs when a user is able to have an application server access or manipulate information from areas that are not normally accessible by the user (such as an internal server). In this case, the application is designed to communicate with external sources, but since we were able to receive data from an internal resource, this counts as an SSRF attack.

Sea-Surf 2

Objective:
Have the application communicate with itself again

Key Information:
The application takes input in the form of an absolute URL, but with restricted addresses.

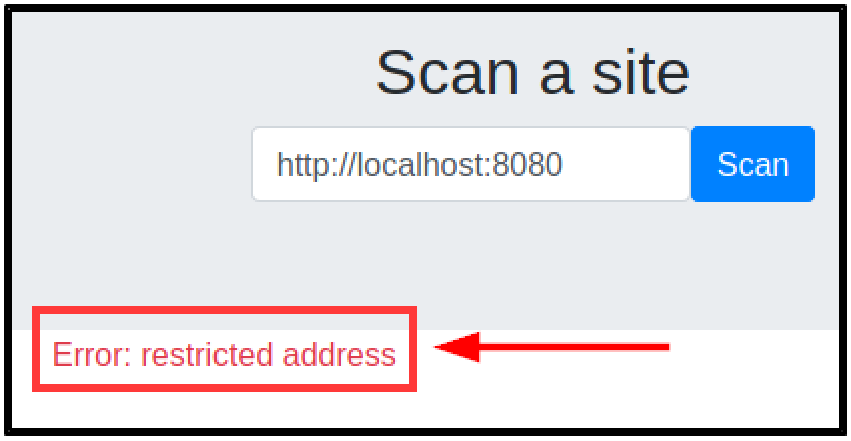
Part 9 – Navigate to the Sea-Surf 2 page:

Step 1 – Go to the following page:

http://ssrf2.local.thetanuki.io/

Step 2 – Attempt the same attack from the previous exercise:

http://localhost:8080

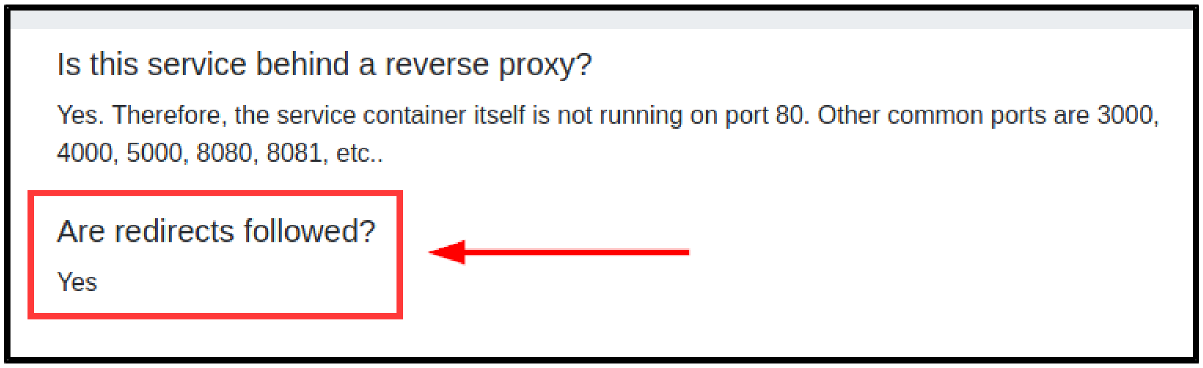


Error: restricted address

Part 10 – Check Out the FAQ Page

Step 1 – Go to the following page:

http://ssrf2.local.thetanuki.io/faq



Is this service behind a reverse proxy?

Yes. Therefore, the service container itself is not running on port 80. Other common ports are 3000, 4000, 5000, 8080, 8081, etc..

Are redirects followed?

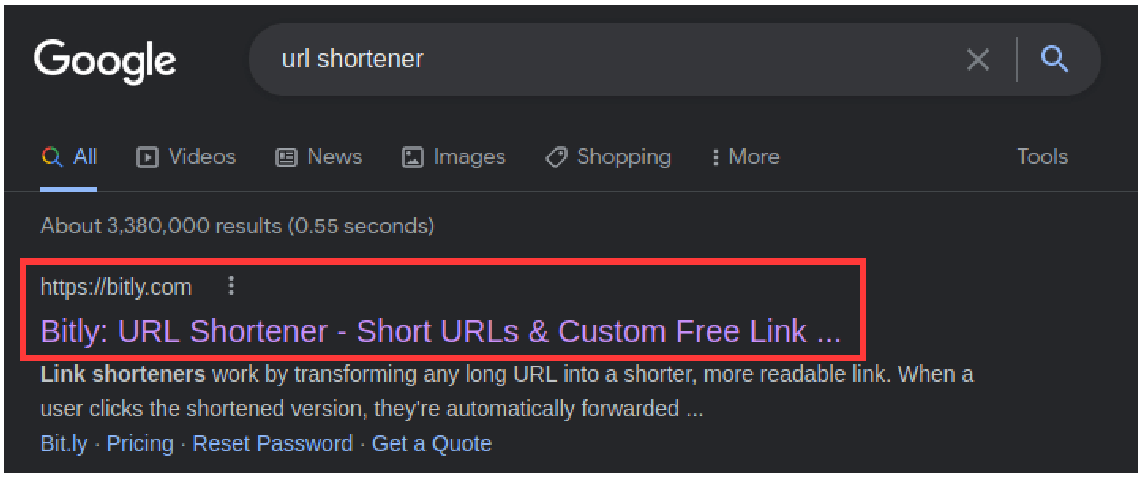
Yes

CONTEXT: The FAQ indicates that URL redirection is allowed by the application, so we can use a URL shortening service as a method of redirecting the address "http://127.0.0.1:8080". "127.0.0.1" is the same as "localhost", in this respect.

Part 11 – Find and Use a URL Shortening Service

Step 1 – Search for the following terms in a search engine:

url shortener



About 3,380,000 results (0.55 seconds)

https://bitly.com

Bitly: URL Shortener - Short URLs & Custom Free Link ...

Link shorteners work by transforming any long URL into a shorter, more readable link. When a user clicks the shortened version, they're automatically forwarded ...

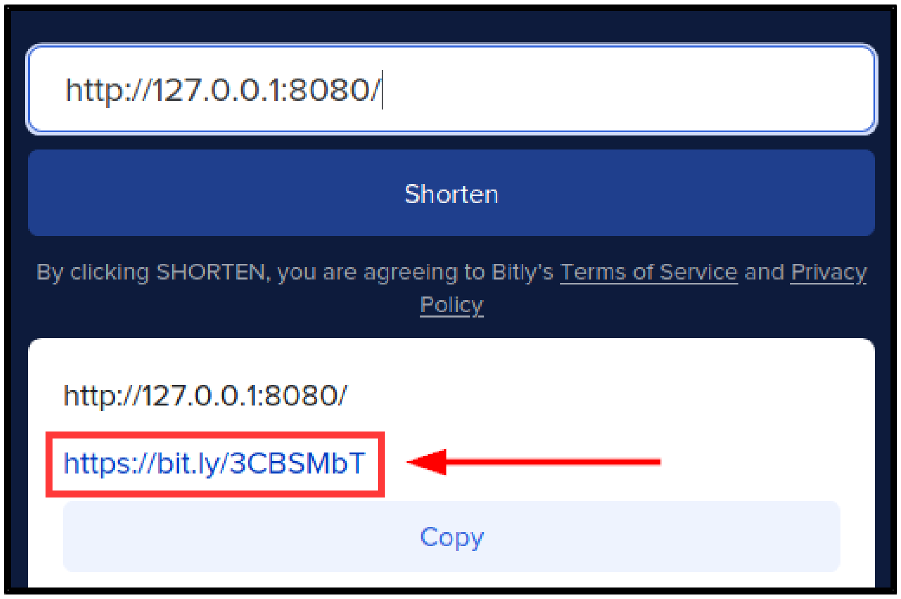
Bit.ly · Pricing · Reset Password · Get a Quote

Step 2 – Navigate to that website:

https://bitly.com

Step 3 – Insert the following the data field, then click Shorten:

http://127.0.0.1:8080/


http://127.0.0.1:8080/

By clicking SHORTEN, you are agreeing to Bitly's Terms of Service and Privacy Policy

http://127.0.0.1:8080/

https://bit.ly/3CBSMbT

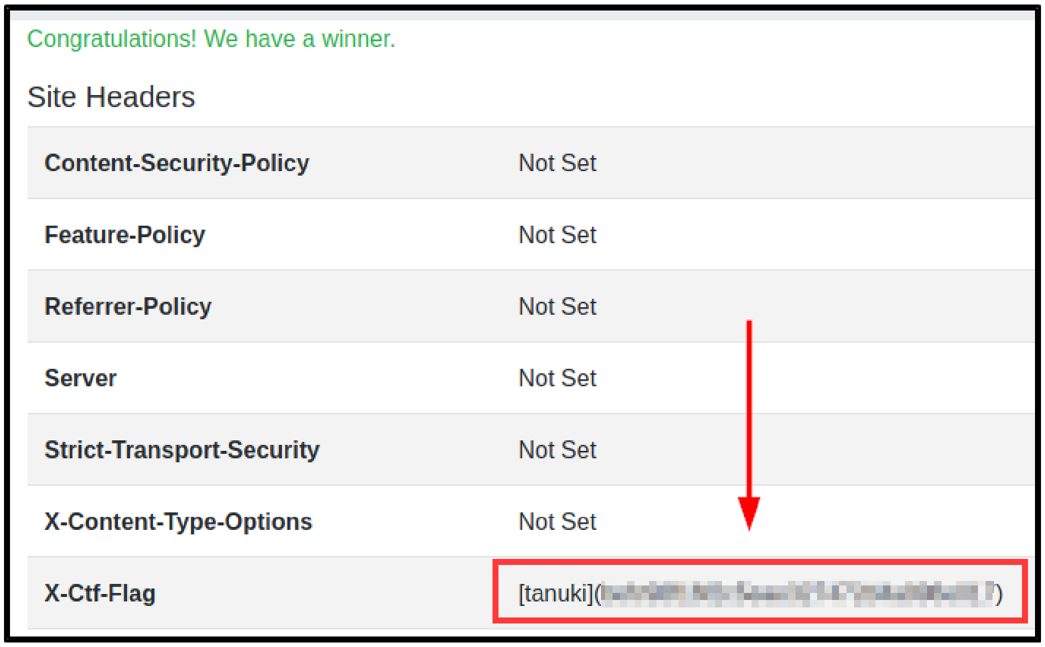
Step 4 – Copy the resulting URL

Part 12 – Enter the Shortened URL into the Sea-Surf Application

Step 1 – Navigate back to the following URL:

http://ssrf2.local.thetanuki.io/

Step 2 – Enter the shortened URL into the application and collect the flag.


Congratulations! We have a winner.

Site Headers

Content-Security-Policy	Not Set
Feature-Policy	Not Set
Referrer-Policy	Not Set
Server	Not Set
Strict-Transport-Security	Not Set
X-Content-Type-Options	Not Set
X-Ctf-Flag	[tanuki]([redacted])

Sea-Surf 3

Objective:
Have the application communicate with itself yet again

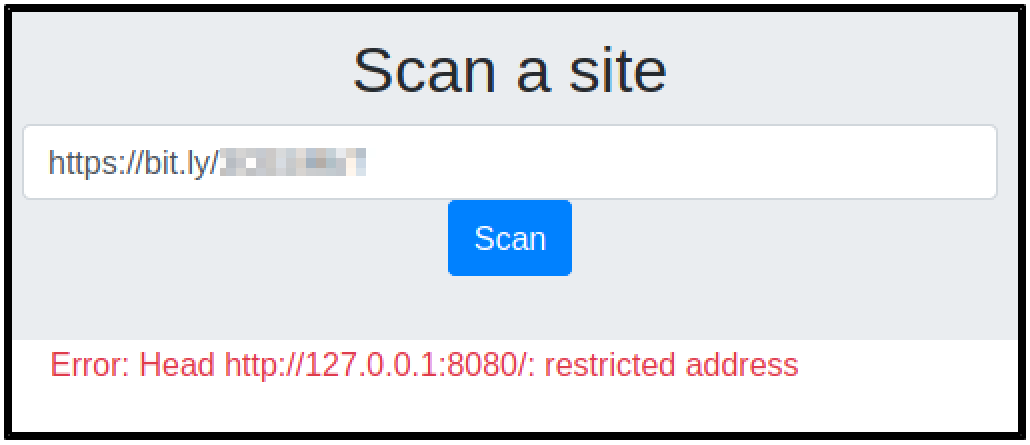
Key Information:
The application takes input in the form of an absolute URL, but with even more restrictions.

Part 13 – Navigate to the Sea-Surf 3 page:

Step 1 – Go to the following page:

http://ssrf3.local.thetanuki.io/

Step 2 – Attempt the same attack from the previous exercise:



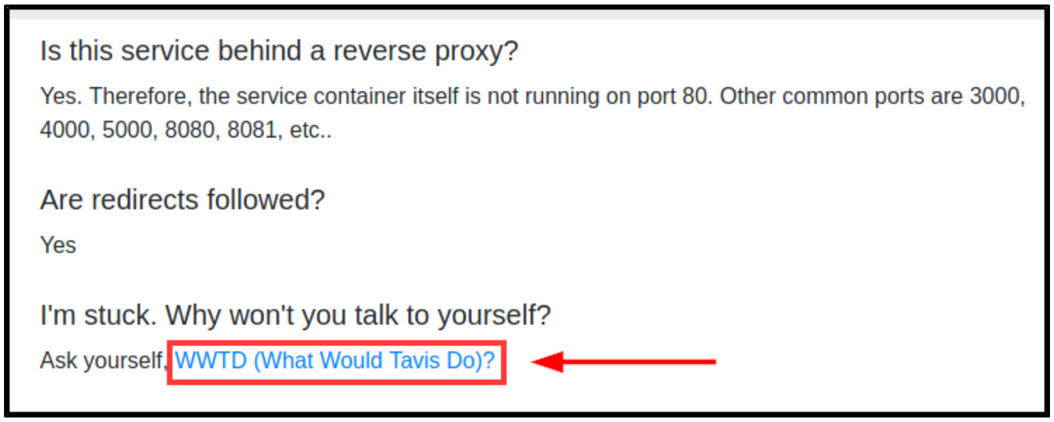
https://bit.ly/[redacted]

Error: Head http://127.0.0.1:8080/: restricted address

Part 14 – Check Out the FAQ Page, Then Follow the Link

Step 1 – Go to the following page:

http://ssrf3.local.thetanuki.io/faq


Is this service behind a reverse proxy?

Yes. Therefore, the service container itself is not running on port 80. Other common ports are 3000, 4000, 5000, 8080, 8081, etc..

Are redirects followed?

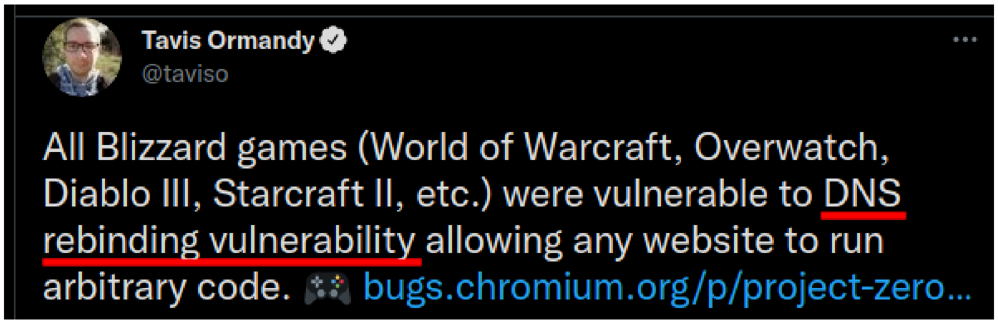
Yes

I'm stuck. Why won't you talk to yourself?

Ask yourself, WWTD (What Would Tavis Do)?

Step 2 – Follow the Link in the FAQ

https://twitter.com/taviso/status/955540415263907840


Tavis Ormandy
@taviso

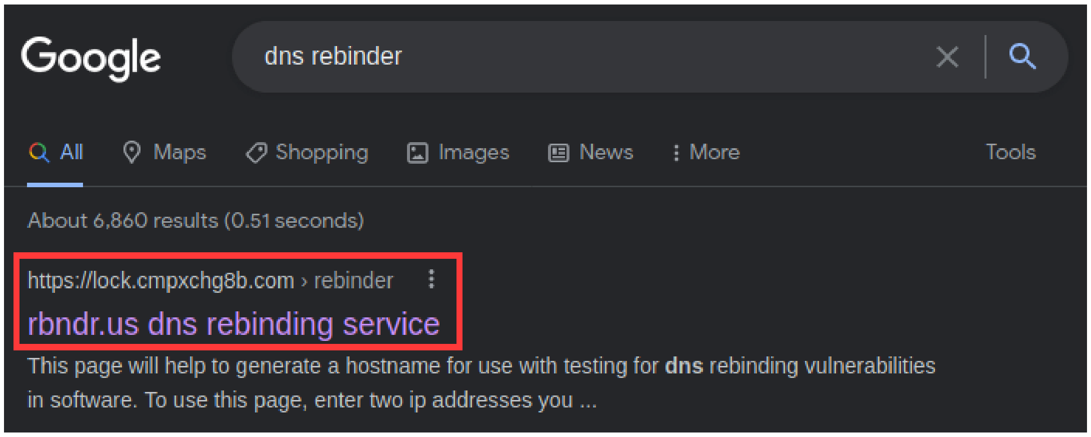
All Blizzard games (World of Warcraft, Overwatch, Diablo III, Starcraft II, etc.) were vulnerable to DNS rebinding vulnerability allowing any website to run arbitrary code. 🎮 bugs.chromium.org/p/project-zero...

CONTEXT: The Twitter post references an exploit called a DNS Rebinding Attack, which allows the manipulation of DNS name resolution and can be used to access endpoints located at private IP addresses.

Part 15 – Find and Configure a DNS Rebinding Service

Step 1 – In a web browser, search for the following terms:

dns rebinder



dns rebinder

About 6,860 results (0.51 seconds)

https://lock.cmpxchg8b.com › rebinder

rbndr.us dns rebinding service

This page will help to generate a hostname for use with testing for **dns** rebinding vulnerabilities in software. To use this page, enter two ip addresses you ...

Step 2 – Navigate to the following website:

https://lock.cmpxchg8b.com/rebinder.html

Step 3 – In the A and B fields, input the following:

A: **8.8.8.8**
B: **127.0.0.1**

Step 4 – Copy the created string

Part 16 – Enter the DNS Rebinded String Into the Application

Step 1 – Return to the Sea-Surf 3 app:

http://ssrf3.local.thetanuki.io/

Step 2 – Enter the DNS rebinded string into the application, prepended with "http://" and appended with ":8080".

http://[dnsRebindedString]**:8080**

NOTE: You may have to send the request several times before the exploit works.

Tar2zip

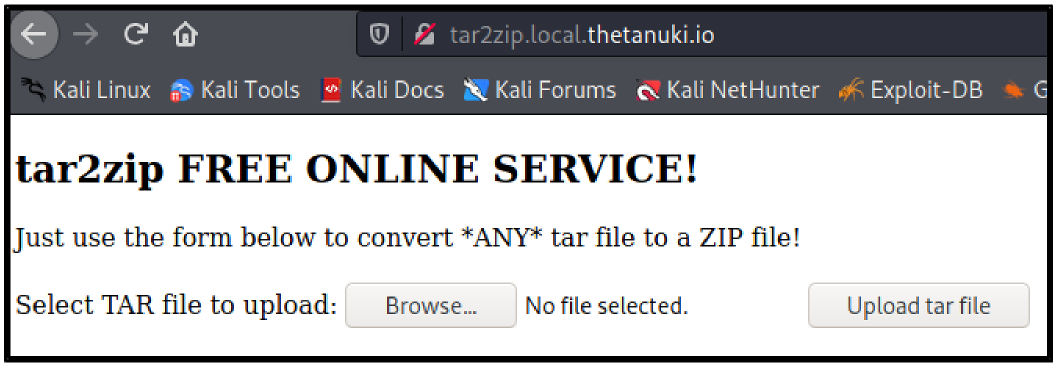
Objective:
Read the flag string in the flag.txt file.

Key Information:
The flag is hidden in **/flag.txt**.

Part 17 – Navigate to the tar2zip page

Step 1 – Go to the following URL:

http://tar2zip.local.thetanuki.io



tar2zip FREE ONLINE SERVICE!

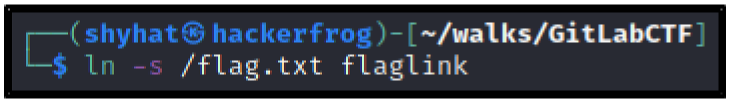
Just use the form below to convert *ANY* tar file to a ZIP file!

Select TAR file to upload: Browse... No file selected. Upload tar file

Part 18 – Create a Symbolic Link File, Then Tar Compress

Step 1 – in the Kali command line, create a symbolic link file to **/flag.txt**:

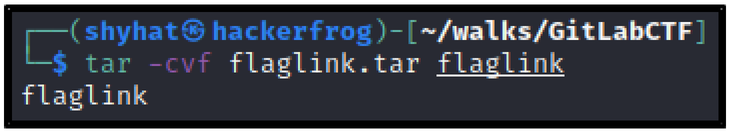
ln -s /flag.txt flaglink



```
┌──(shyhat㉿hackerfrog)-[~/walks/GitLabCTF]
└─$ ln -s /flag.txt flaglink
```

Step 2 – tar compress the file created in Step 1.

tar cvf flaglink.tar flaglink



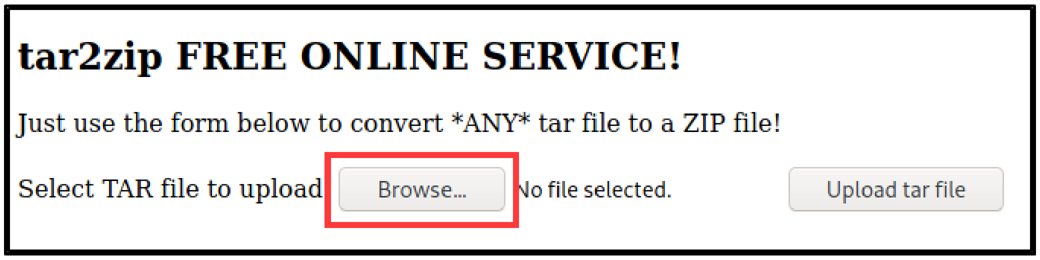
```
┌──(shyhat㉿hackerfrog)-[~/walks/GitLabCTF]
└─$ tar -cvf flaglink.tar flaglink
flaglink
```

Part 19 – Upload The Tar File to the Application, then Download the Converted File

Step 1 – upload the tar file to the application:

click **Browse**
navigate to the tar file, and select it, then click Open

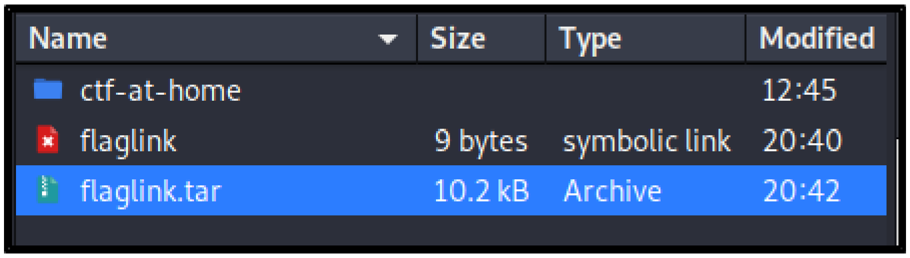
click the **Upload** button



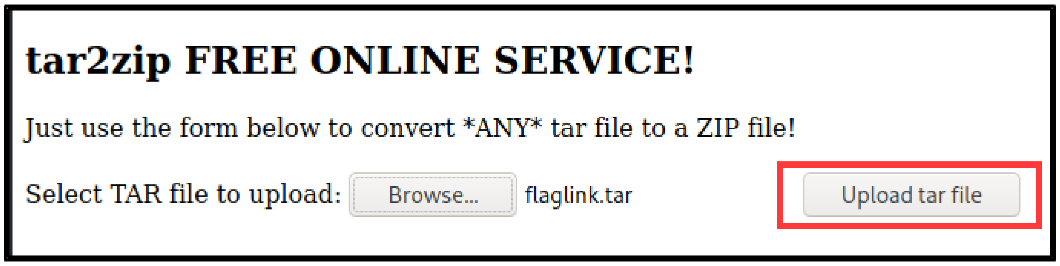
tar2zip FREE ONLINE SERVICE!

Just use the form below to convert *ANY* tar file to a ZIP file!

Select TAR file to upload: Browse... No file selected. Upload tar file



Name	Size	Type	Modified
ctf-at-home			12:45
flaglink	9 bytes	symbolic link	20:40
flaglink.tar	10.2 kB	Archive	20:42



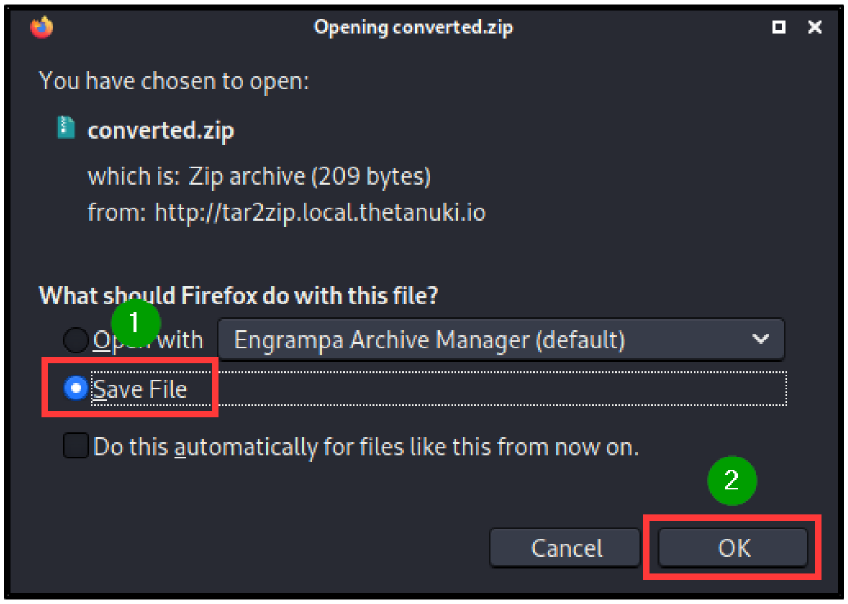
tar2zip FREE ONLINE SERVICE!

Just use the form below to convert *ANY* tar file to a ZIP file!

Select TAR file to upload: Browse... flaglink.tar Upload tar file

Step 2 – Download the converted file

click Save File
click Okay



Opening converted.zip

You have chosen to open:

converted.zip

which is: Zip archive (209 bytes)
from: http://tar2zip.local.thetanuki.io

What should Firefox do with this file?

Open with [ ] Engrampa Archive Manager (default)

[✓ Save File]

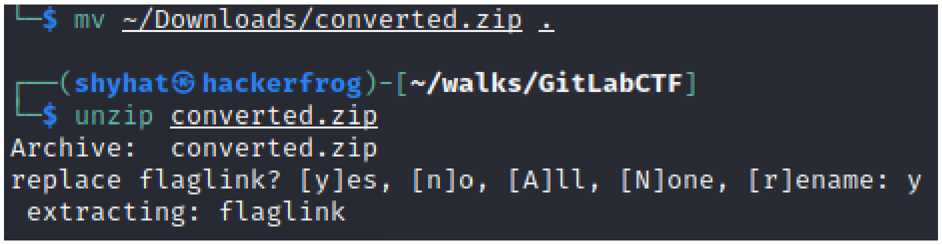
[ ] Do this automatically for files like this from now on.

Cancel OK

Part 20 – Obtain the Flag String From the Zip File

Step 1 – Move the Zip file from the Downloads folder to your working directory, then unzip the file:

mv ~/Downloads/flaglink.zip .
unzip flaglink.zip

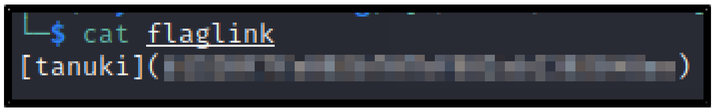


```
└─$ mv ~/Downloads/converted.zip .

┌──(shyhat㉿hackerfrog)-[~/walks/GitLabCTF]
└─$ unzip converted.zip
Archive:  converted.zip
replace flaglink? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
 extracting: flaglink
```

Step 2 – read the flag in the resulting file

cat flaglink



```
└─$ cat flaglink
[tanuki]([redacted])
```

CONTEXT

Because the file we created is a hard-linked file to /flag.txt, the contents of that file on the webserver will be copied when the link file is extracted from the tar file.

GTP

Objective:
Determine the password for the **admin** user account and login to the application.

Key Information:
The source-code for the application is available.

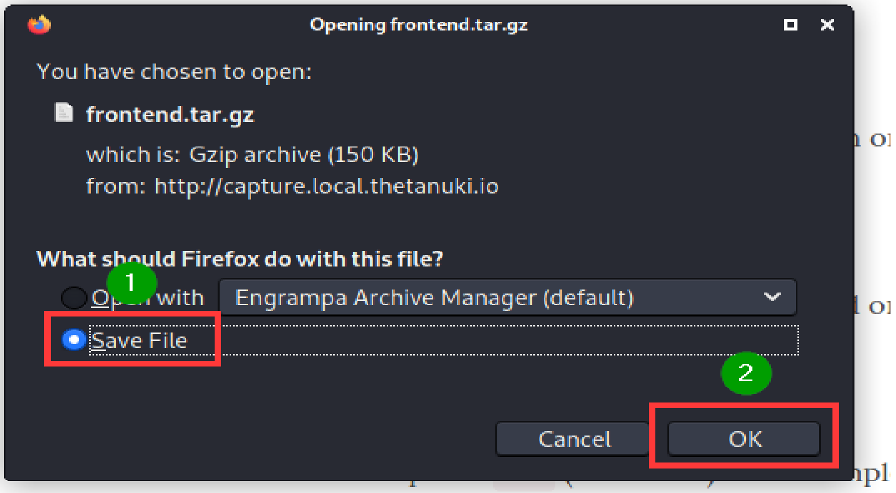
Part 21 – Access the GTP/OTP Challenge Page and Download the Source-code

Step 1- Access the GTP challenge page at the following address:

http://capture.local.thetanuki.io/gtp-otp/

Step 2 – Download the GTP challenge source-code:

http://capture.local.thetanuki.io/frontend.tar.gz



Opening frontend.tar.gz

You have chosen to open:

frontend.tar.gz

which is: Gzip archive (150 KB)
from: http://capture.local.thetanuki.io

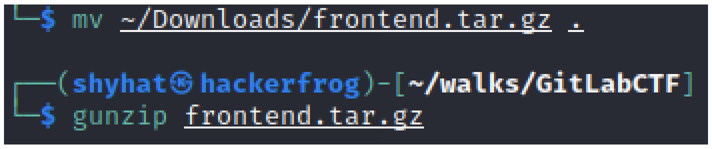
What should Firefox do with this file?

Open with Engrampa Archive Manager (default) [✓ Save File]

Part 22 – Move and Extract the Source-code

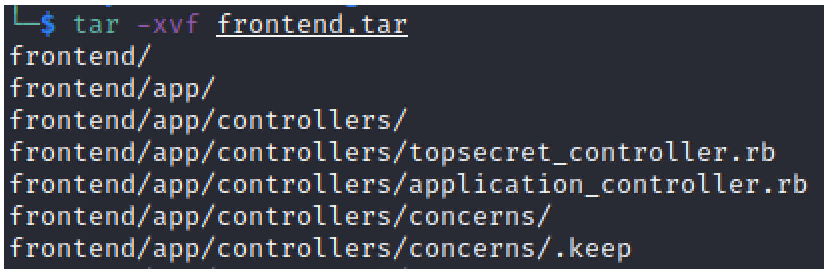
Step 1 – Move the source-code archive file to your working directory and decompress the file twice:

```
mv ~/Downloads/frontend.tar.gz .
gunzip frontend.tar.gz
tar -xvf frontend.tar
```



```
└─$ mv ~/Downloads/frontend.tar.gz .

┌──(shyhat㉿hackerfrog)-[~/walks/GitLabCTF]
└─$ gunzip frontend.tar.gz
```

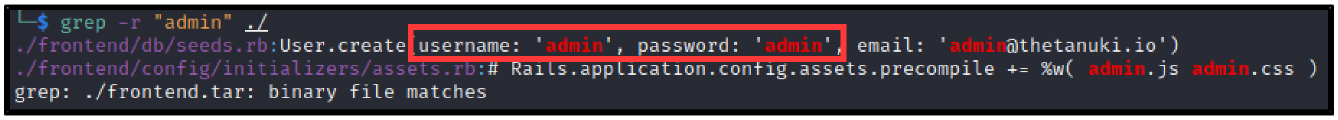


```
└─$ tar -xvf frontend.tar
frontend/
frontend/app/
frontend/app/controllers/
frontend/app/controllers/topsecret_controller.rb
frontend/app/controllers/application_controller.rb
frontend/app/controllers/concerns/
frontend/app/controllers/concerns/.keep
```

Part 23 – Search the Source-code Files for Admin Credentials

Step 1 – Use grep to search the frontend source-code for the term "admin".

grep -r "admin" ./

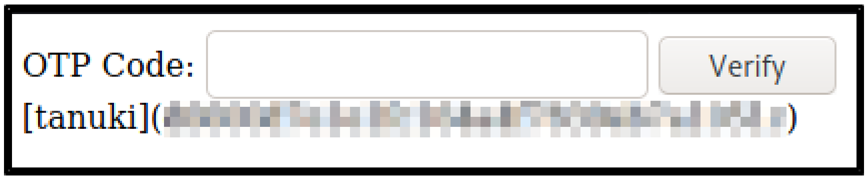


```
└─$ grep -r "admin" ./
./frontend/db/seeds.rb:User.create(username: 'admin', password: 'admin', email: 'admin@thetanuki.io')
./frontend/config/initializers/assets.rb:# Rails.application.config.assets.precompile += %w( admin.js admin.css )
grep: ./frontend.tar: binary file matches
```

Part 24 – Login to the Application Using the Found Credentials

Step 1- Access the application login page and log in:

http://otp.local.thetanuki.io/users/sign_in
username: **admin**
password: **admin**



OTP Code: Verify
[tanuki]([redacted])