EKOPARTY 2020

PWNDEMIC

Advanced fuzzing workshop

Antonio Morales

English & Spanish friendly

- Key concepts in both languages
- You can ask me anything (ENG/ES)

- Los conceptos importantes se explicarán en ambos idiomas.
- Me puedes preguntar en cualquiera de los 2 idiomas

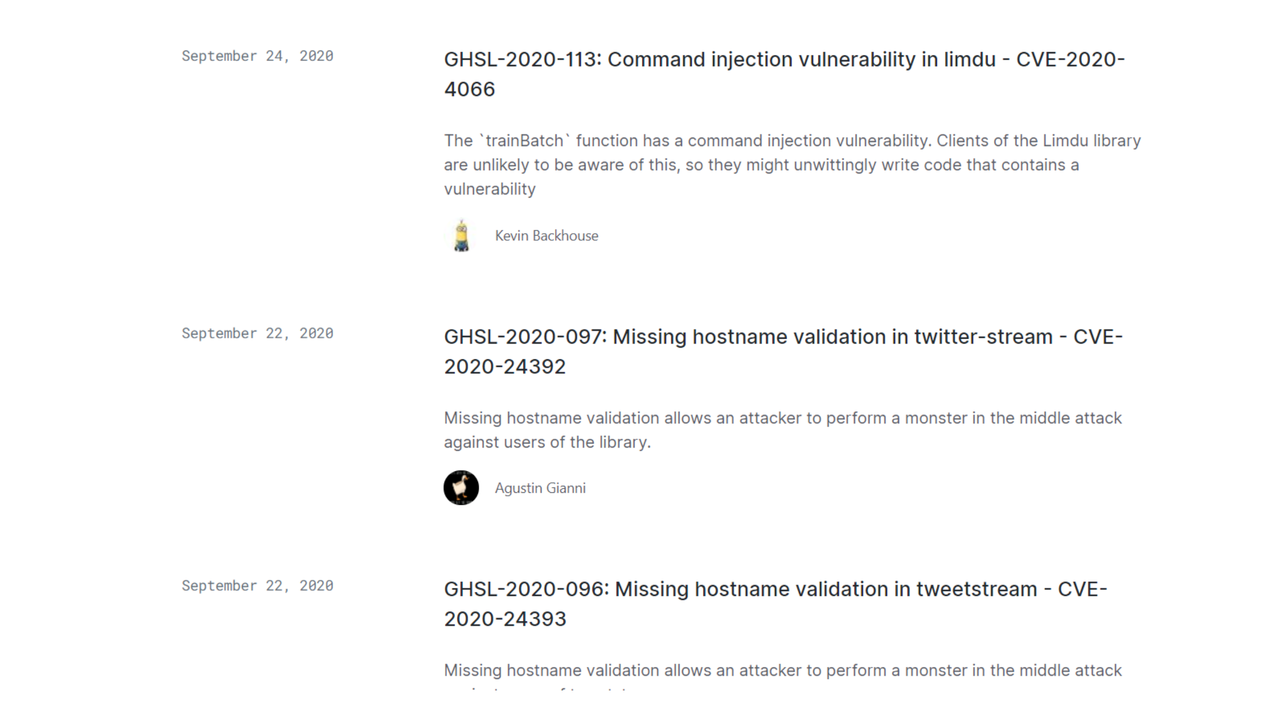
// WHO AM I

#define speaker	Antonio Morales
#define job	Security Researcher at GitHub
#define twitter	@nosoynadiemas

using namespace EkoParty;

int main(int argc, char argv[]){*

Security Lab



September 24, 2020

GHSL-2020-113: Command injection vulnerability in limdu - CVE-2020-4066

The `trainBatch` function has a command injection vulnerability. Clients of the Limdu library are unlikely to be aware of this, so they might unwittingly write code that contains a vulnerability

Kevin Backhouse

September 22, 2020

GHSL-2020-097: Missing hostname validation in twitter-stream - CVE-2020-24392

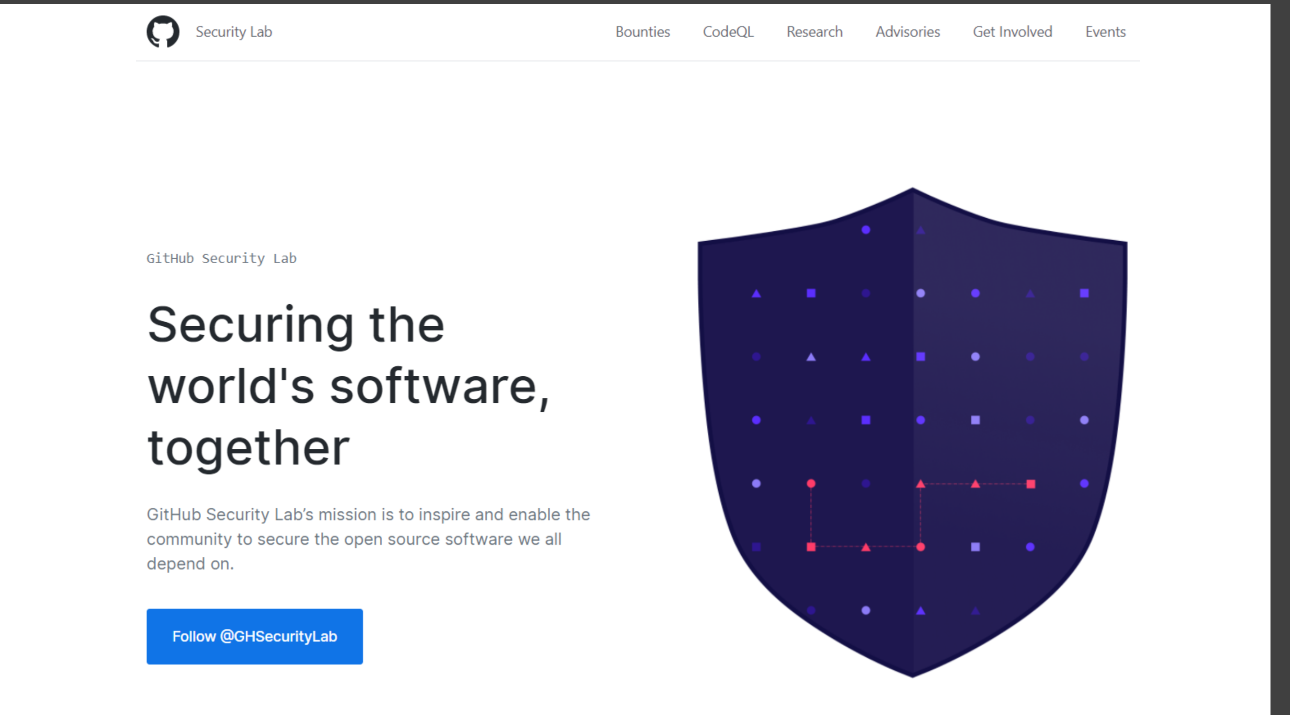
Missing hostname validation allows an attacker to perform a monster in the middle attack against users of the library.

Agustin Gianni

September 22, 2020

GHSL-2020-096: Missing hostname validation in tweetstream - CVE-2020-24393

Missing hostname validation allows an attacker to perform a monster in the middle attack

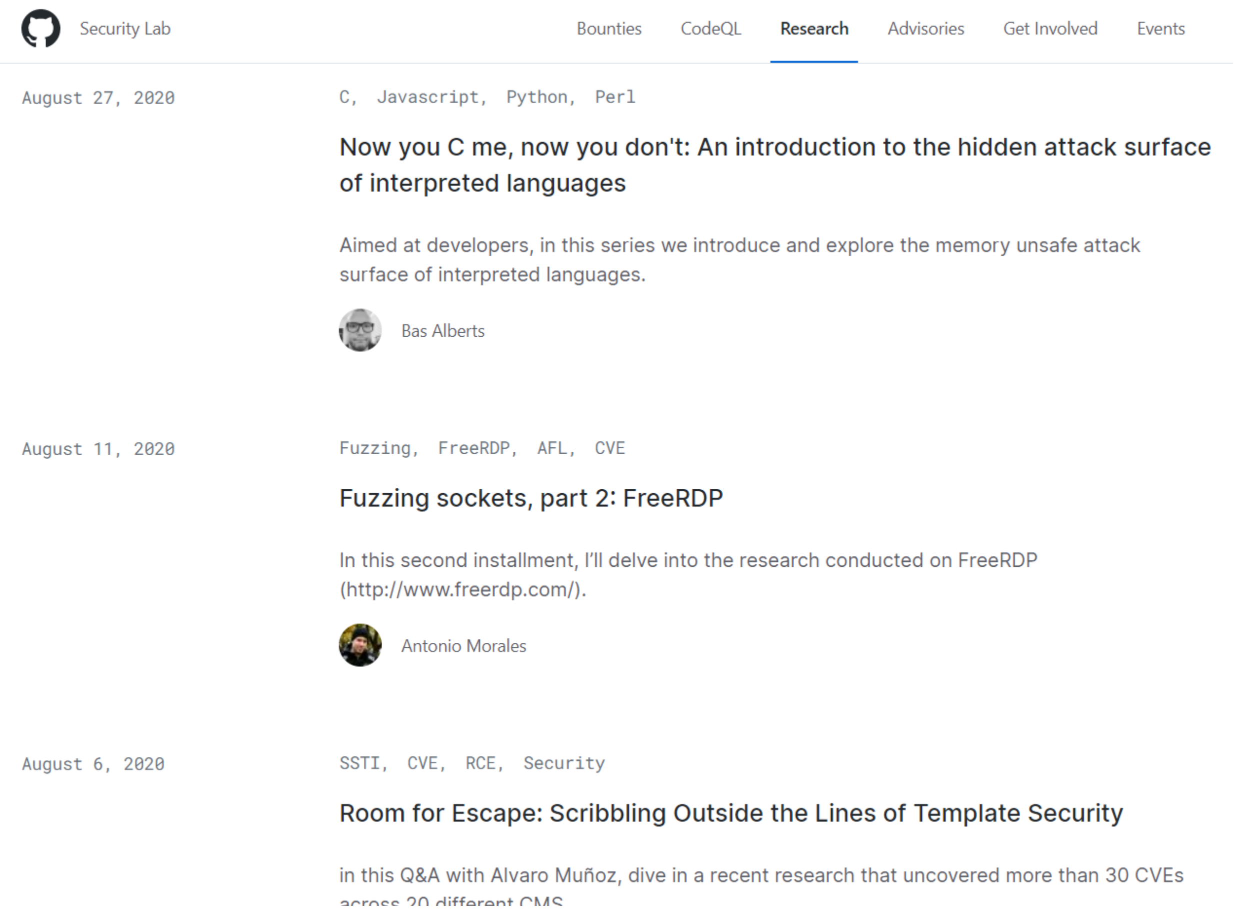


GitHub Security Lab

Securing the world's software, together

GitHub Security Lab's mission is to inspire and enable the community to secure the open source software we all depend on.

https://securitylab.github.com/



August 27, 2020

C, Javascript, Python, Perl

Now you C me, now you don't: An introduction to the hidden attack surface of interpreted languages

Aimed at developers, in this series we introduce and explore the memory unsafe attack surface of interpreted languages.

Bas Alberts

August 11, 2020

Fuzzing, FreeRDP, AFL, CVE

Fuzzing sockets, part 2: FreeRDP

In this second installment, I'll delve into the research conducted on FreeRDP (http://www.freerdp.com/).

Antonio Morales

August 6, 2020

SSTI, CVE, RCE, Security

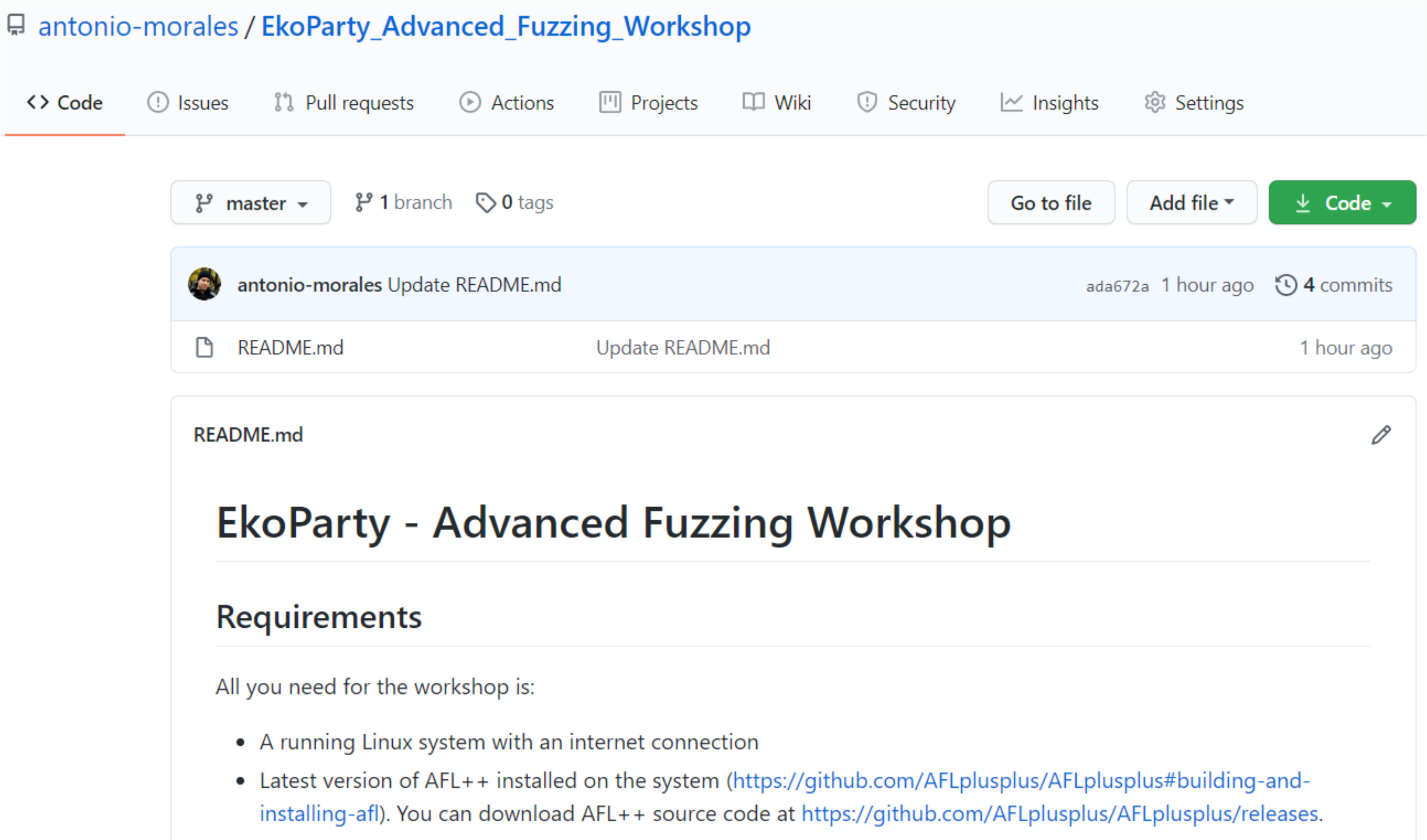
Room for Escape: Scribbling Outside the Lines of Template Security

in this Q&A with Alvaro Muñoz, dive in a recent research that uncovered more than 30 CVEs across 20 different CMS

@GHSecurityLab

Workshop repository

There you can find all you need for the workshop: https://github.com/antonio-morales/EkoParty_Advanced_Fuzzing_Workshop



antonio-morales Update README.md ada672a 1 hour ago 4 commits

README.md Update README.md 1 hour ago

README.md

EkoParty - Advanced Fuzzing Workshop

Requirements

All you need for the workshop is:

- A running Linux system with an internet connection
- Latest version of AFL++ installed on the system (https://github.com/AFLplusplus/AFLplusplus#building-and-installing-afl). You can download AFL++ source code at https://github.com/AFLplusplus/AFLplusplus/releases.

Motivation

CVE-2019-20176	CVE-2019-14438	CVE-2019-14777	CVE-2020-4030	CVE-2020-9273
CVE-2020-9274	CVE-2019-14498	CVE-2019-14970	CVE-2020-11096	CVE-2019-14778
CVE-2020-9365	CVE-2019-14535	CVE-2020-13396	CVE-2020-11095	CVE-2020-11097
CVE-2020-6162	CVE-2019-14534	CVE-2020-13397	CVE-2020-4032	CVE-2019-14437
CVE-2020-6835	CVE-2019-14533	CVE-2020-13398	CVE-2020-4033	CVE-2019-14779
CVE-2020-9272	CVE-2019-14776	CVE-2020-11099	CVE-2020-4031	CVE-2020-11098

The aim of this workshop

Dumb Fuzzing

Smart Fuzzing

Workshop Format

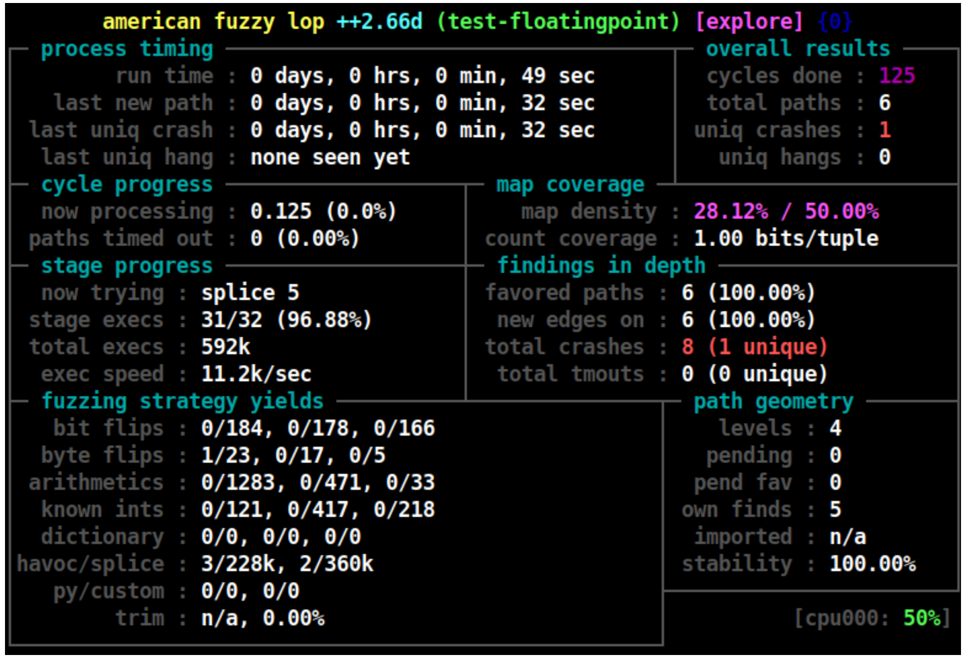
- It’s a hands-on CTF-style workshop (learning-by-doing method).
- You will learn while facing the challenges. I’m here to guide your learning.

- Es un taller totalmente práctico (basado en el aprendizaje autónomo)
- Aprenderás a través de intentar los retos. Mi labor será la de guiar tu aprendizaje.

Tools

All you need for the workshop is **AFL++ tool** running on a Linux system. Please, if you haven't download yet, do it now: https://github.com/AFLplusplus/AFLplusplus/releases

Installing AFL++ -> https://github.com/AFLplusplus/AFLplusplus#building-and-installing-afl



```
       american fuzzy lop ++2.66d (test-floatingpoint) [explore] {0}
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 0 min, 49 sec      │  cycles done : 125   │
│   last new path : 0 days, 0 hrs, 0 min, 32 sec      │  total paths : 6     │
│ last uniq crash : 0 days, 0 hrs, 0 min, 32 sec      │ uniq crashes : 1     │
│  last uniq hang : none seen yet                     │   uniq hangs : 0     │
├─ cycle progress ────────────────────┬─ map coverage ─┴──────────────────────┤
│  now processing : 0.125 (0.0%)      │    map density : 28.12% / 50.00%      │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.00 bits/tuple      │
├─ stage progress ────────────────────┼─ findings in depth ───────────────────┤
│  now trying : splice 5              │ favored paths : 6 (100.00%)           │
│ stage execs : 31/32 (96.88%)        │  new edges on : 6 (100.00%)           │
│ total execs : 592k                  │ total crashes : 8 (1 unique)          │
│  exec speed : 11.2k/sec             │  total tmouts : 0 (0 unique)          │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 0/184, 0/178, 0/166                   │    levels : 4          │
│  byte flips : 1/23, 0/17, 0/5                       │   pending : 0          │
│ arithmetics : 0/1283, 0/471, 0/33                   │  pend fav : 0          │
│  known ints : 0/121, 0/417, 0/218                   │ own finds : 5          │
│  dictionary : 0/0, 0/0, 0/0                         │  imported : n/a        │
│havoc/splice : 3/228k, 2/360k                        │ stability : 100.00%    │
│   py/custom : 0/0, 0/0                              ├────────────────────────┘
│        trim : n/a, 0.00%                            │          [cpu000: 50%]
└─────────────────────────────────────────────────────┘
```

Para el workshop todo lo que necesitas es AFL++ . Si aún no lo has descargado, hazlo ahora: https://github.com/AFLplusplus/AFLplusplus/releases

Como instalar AFL++ -> https://github.com/AFLplusplus/AFLplusplus#building-and-installing-afl

RULES

Rule 1

- Challenges are intended to be solved by fuzzing.
- But you can use whatever method you want (good luck xD)

- Las pruebas están pensadas para ser resultas mediante fuzzing.
- Pero puedes utilizar el método que desees (buena suerte xD)

Rule 2

- There will be **3 different challenges**. The goal is to **find a reproducible bug** on each of them.
- We're looking for exploitable vulnerabilities. "Theoretical bugs" or code warnings are not welcome, sorry.
- In order to be the winner of a challenge, **you must provide a crash/PoC**.

- Habrá **3 pruebas distintas**. El objetivo es encontrar un bug en cada una de ellas.
- Se trata de encontrar vulnerabilidades explotables. Bugs teóricos o alertas de código no son bienvenidas. Además, para ser ganador del reto deberás de entregar un crash or PoC.

Rule 3

- Please, don’t disclose your solutions.
- Upload them to Google Drive / Dropbox / Onedrive or whatever cloud storage tool, and **send me the link via private message.**

- Por favor, no reveles tus soluciones.
- En su lugar, subelas a Google Drive / Dropbox / Onedrive o cualquier servidor en la nube y **envíame por privado el enlace**

Rule 4

- I will give you some hints and tips before and during the challenge.
- I'll release a **new hint every 10 minutes** (approx.)

- Daré varios consejos y pistas antes y durante cada reto
- Liberaré una **nueva pista cada 10 minutos** aproximadamente

Rule 5

- After each challenge, I will show my solution and I will explain it to you.
- There may be more than one correct solution.

- Daré varios consejos y pistas antes y durante cada reto
- Liberaré una **nueva pista cada 10 minutos** aproximadamente

Awards

- There will be **2 winners for each challenge** (6 total winners).
- The winners will be the fastest ones in solving the challenge (find the vulnerability).

- Cada reto tendrá 2 ganadores (6 ganadores total)
- Los ganadores serán los más rápidos en resolver el reto (encontrar la vulnerabilidad).

Prizes

GitHub SHOP

Sold Out

Invertocat 2.0 Shirt
$25.00

Arctocat Shirt
$25.00

I [octocat] Code 2.0 Shirt
$25.00

GitHub Username Shirt
$25.00

Atom Shirt
$25.00

Atom 2.0 Shirt
$25.00

Octocat One-Piece
$18.00

Kids Octocat Raglan Tee
$18.00

GitHub Drip Tee
$25.00

Questocat Tee
$25.00

Invertocat 3.0 Shirt
$25.00

Sold Out

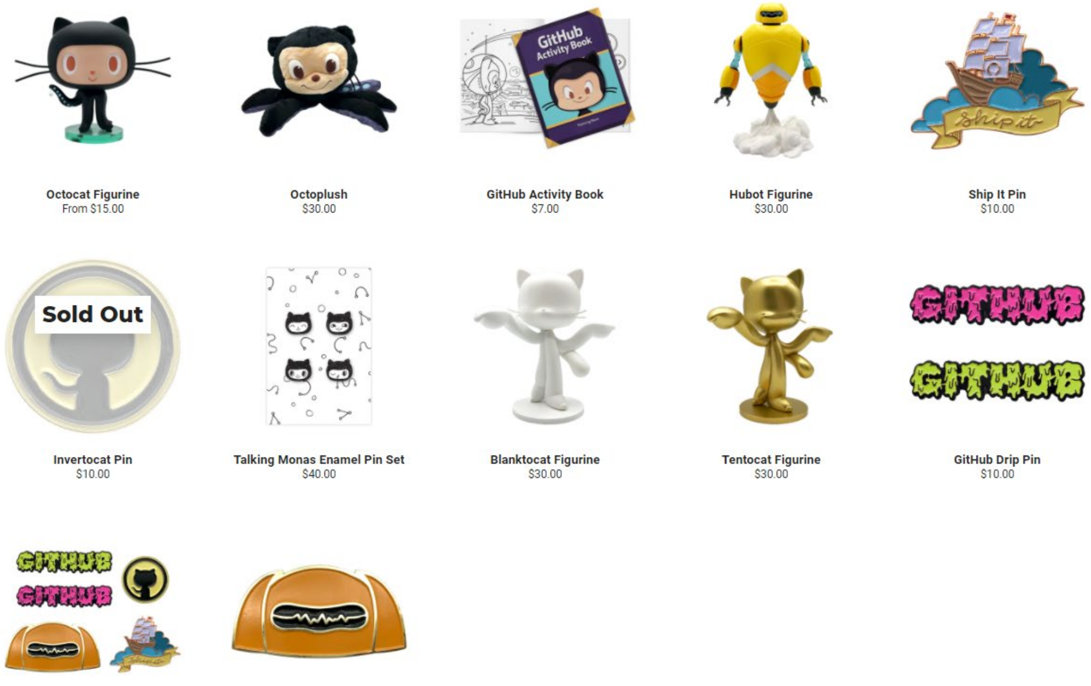
De Los Muertos Shirt
$25.00

Grim Repo Shirt
$25.00

Sold Out

Talking Monas - Kid's Raglan
$18.00

Talking Monas - Onesie
$18.00



Octocat Figurine
From $15.00

Octoplush
$30.00

GitHub Activity Book
$7.00

Hubot Figurine
$30.00

Ship It Pin
$10.00

Sold Out

Invertocat Pin
$10.00

Talking Monas Enamel Pin Set
$40.00

Blanktocat Figurine
$30.00

Tentocat Figurine
$30.00

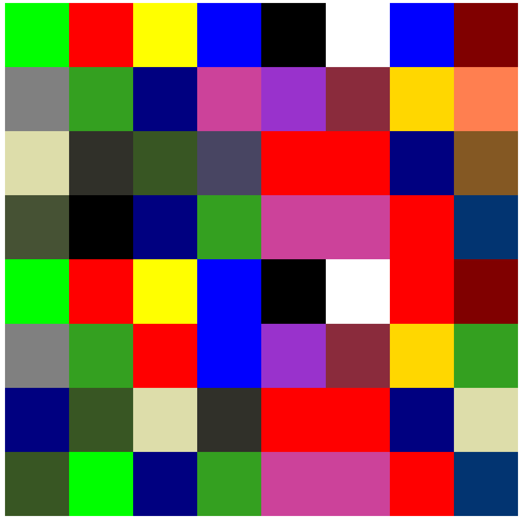
GitHub Drip Pin
$10.00

https://github.myshopify.com/

QUESTIONS / PREGUNTAS

Challenge 1 - ESIF (Extremely Stupid Image Format)

Get the code at: https://github.com/antonio-morales/EkoParty_Advanced_Fuzzing_Workshop/



Convert ESIF format to PPM format

Build:

```
> gcc EkoParty1.c -o EkoParty1 -w -lcrypto -lssl
```

Run:

```
>  ./EkoParty1 example.ESIF output.ppm
```

You can find "Example.ESIF" in the repository

Puedes encontrar "Example.ESIF" en el repositorio

Challenge 1 - ESIF (Extremely Stupid Image Format)

Ask me any doubt
via PM

Reminder

40 minutes

LET'S GO!!!

Challenge 1 – Tip

- I strongly advise you to link your binary with **ASan (AddressSanitizer)** and **UBSan (Undefined Behavior Sanitizer)**
- To do this, add **-fsanitize=address,undefined** to your compile line
- Don’t forget to add **-m none** to your AFL command line

- Te aconsejo encarecidamente que enlaces tu binario con **ASan (AddressSanitizer)** y **UBSan (Undefined Behavior Sanitizer)**
- Para ello, añade **-fsanitize=address,undefined** a tu linea de compilación
- No te olvides de añadir **–m none** a tu línea de comandos de AFL

Challenge 1 – Hint 1

- Code coverage can be really useful here.
- You can enable it adding **--coverage** to your compile line
- I've just uploaded a Code Coverage folder to the repo2 new files to the repo: **lcov.sh** and **run_files**
- You can collect code coverage, as follows:

```
>  chmod +x run_files
>  chmod +x lcov.sh
> ./lcov.sh
```

Then, open **./html_coverage/index.html** to view generated LCOV code coverage report

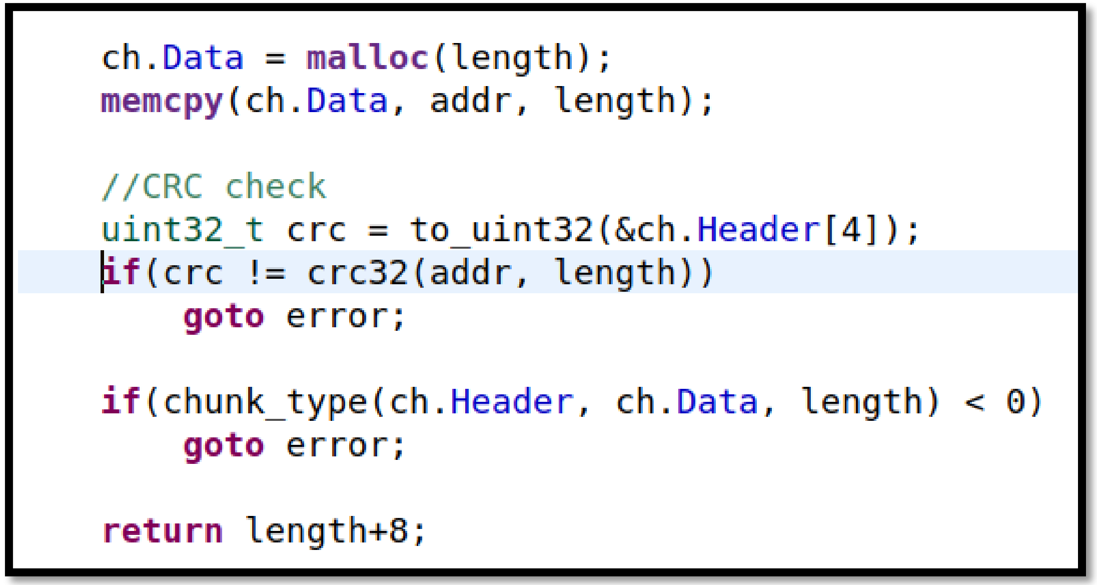
Challenge 1 – Hint 2

- Sometimes checksums can be a pain in the ass.
- Take a look at: https://securitylab.github.com/research/fuzzing-challenges-solutions-1

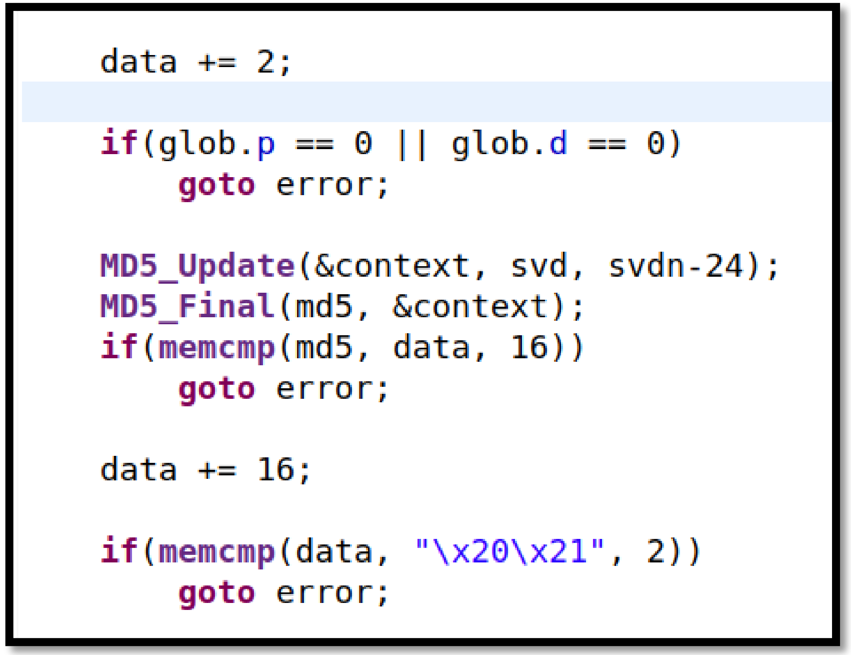
- En ocasiones los checksums pueden ser realmente molestos
- Echa un vistazo a: https://securitylab.github.com/research/fuzzing-challenges-solutions-1

Challenge 1 – Hint 3

Looks like there are some obstacles in the code...



```
ch.Data = malloc(length);
memcpy(ch.Data, addr, length);

//CRC check
uint32_t crc = to_uint32(&ch.Header[4]);
if(crc != crc32(addr, length))
    goto error;

if(chunk_type(ch.Header, ch.Data, length) < 0)
    goto error;

return length+8;
```



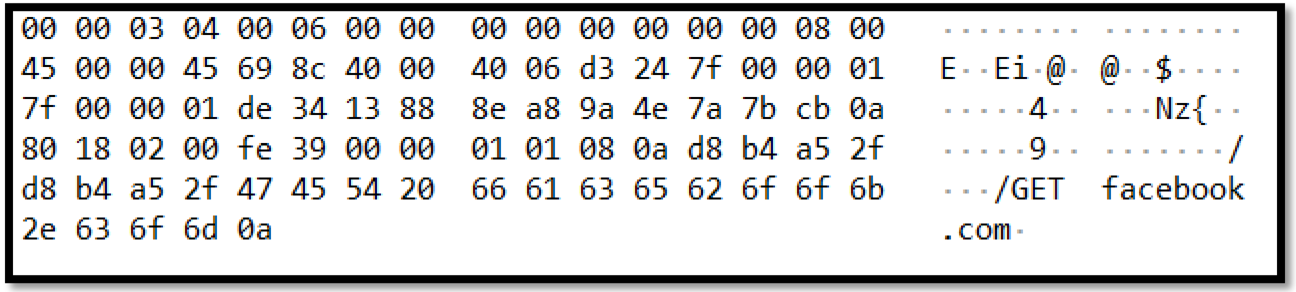
```
data += 2;

if(glob.p == 0 || glob.d == 0)
    goto error;

MD5_Update(&context, svd, svdn-24);
MD5_Final(md5, &context);
if(memcmp(md5, data, 16))
    goto error;

data += 16;

if(memcmp(data, "\x20\x21", 2))
    goto error;
```

Parece que hay algunos obstáculos en el código...

Challenge 1 – My Solution

Challenge 2 – Crazy HTTP Server

Get the code at: https://github.com/antonio-morales/EkoParty_Advanced_Fuzzing_Workshop/



```
00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
45 00 00 45 69 8c 40 00  40 06 d3 24 7f 00 00 01   E··Ei·@· @··$····
7f 00 00 01 de 34 13 88  8e a8 9a 4e 7a 7b cb 0a   ·····4·· ···Nz{··
80 18 02 00 fe 39 00 00  01 01 08 0a d8 b4 a5 2f   ·····9·· ·······/
d8 b4 a5 2f 47 45 54 20  66 61 63 65 62 6f 6f 6b   ···/GET  facebook
2e 63 6f 6d 0a                                     .com·
```

An HTTP Server that is not what it seems!

Build:

```
> gcc EkoParty2.c -o EkoParty2 -w -lz
```

Run (as root):

```
> ./EkoParty2
```

You can find some capture examples in the “Captures” folder

Puedes encontrar algunos ejemplos de paquetes capturados en el directorio “Captures”

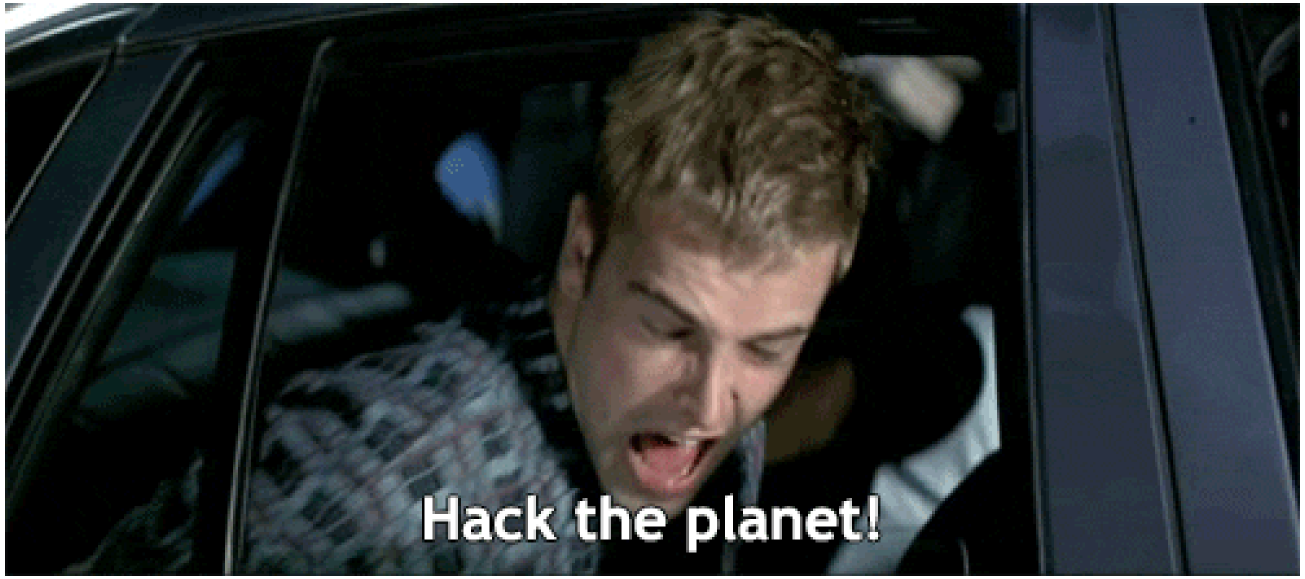
Challenge 2 - Crazy HTTP Server

Ask me any doubt
via PM

Reminder

50 minutes

LET’S GO!!!



Challenge 2 - Tip

- A **dictionary** can be useful… sometimes
- afl-fuzz command:

  ```
  afl-fuzz -t 500 -m none -i ../AFL/afl_in/ -o ../AFL/afl_out -x ../AFL/mydict.txt  -- ./EkoParty2 @@
  ```

If you need more help, take a look at: https://securitylab.github.com/research/fuzzing-challenges-solutions-1 *("Providing a custom dictionary")*

- En ocasiones un **diccionario** puede ser util
- afl-fuzz command:

  ```
  afl-fuzz -t 500 -m none -i ../AFL/afl_in/ -o ../AFL/afl_out -x ../AFL/mydict.txt  -- ./EkoParty2 @@
  ```

Si necesitas mas ayuda, echa un vistazo a: https://securitylab.github.com/research/fuzzing-challenges-solutions-1 *("Providing a custom dictionary")*

Challenge 2 – Hint 1

- The TCP/IP **port numbers below 1024** are special in that normal users are not allowed to run servers on them.

- Maybe you can change this port

- Los puertos TCP/IP por debajo de 1024 son privilegiados de forma que un usuario con privilegios normales no pueda ejecutar un servidor en ellos

- Quizás puedas cambiar el puerto

Challenge 2 – Hint 2

- Have you been able to extract the .PCAP content?
- If not, now you can download the raw content from GitHub repository

---

- Has podido extraer el contenido de los archivos .PCAP?
- Si no, puedes descargarte el contenido extraido del repositorio de GitHub

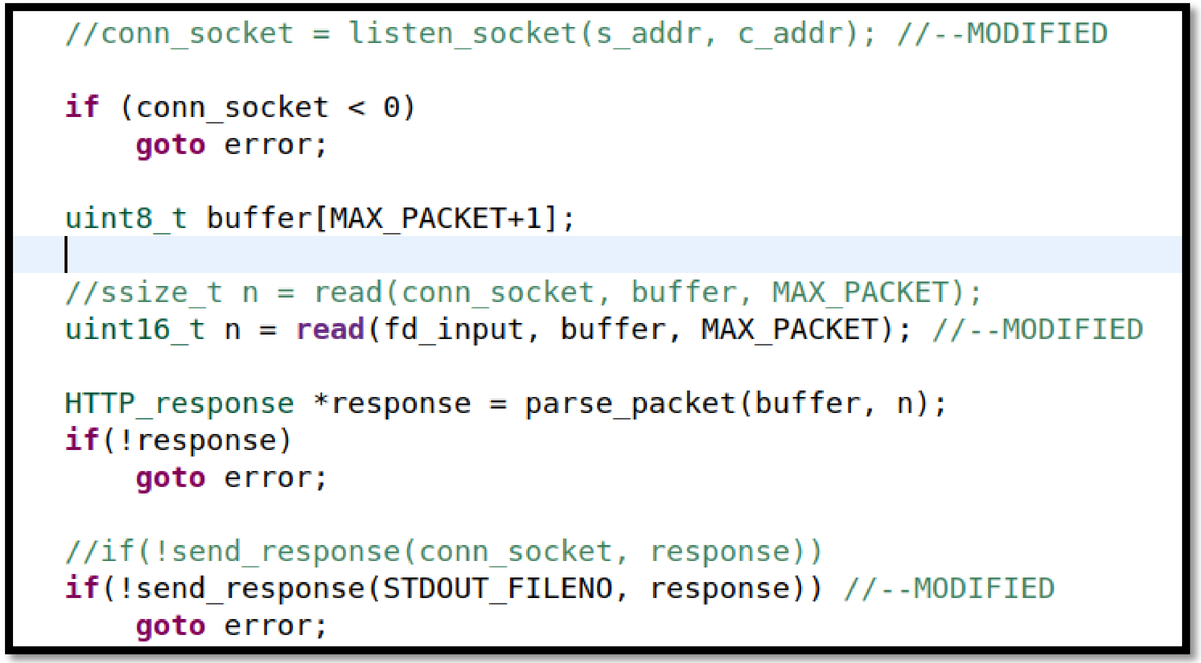
Challenge 2 – Hint 3

- AFL doesn’t support sockets natively. Maybe this link could help you: https://securitylab.github.com/research/fuzzing-sockets-FTP

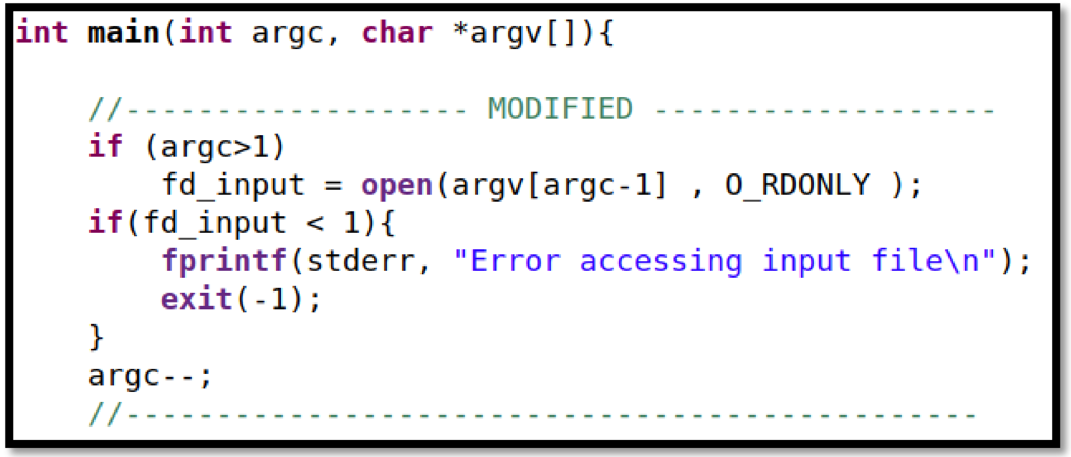
- AFL no soporta de forma nativa el fuzzeo de sockets. Pero quizás este link te pueda ser de ayuda: https://securitylab.github.com/research/fuzzing-sockets-FTP

Challenge 2 – Hint 4

Still not successful fuzzing sockets? Ok, look these code snippets



```
//conn_socket = listen_socket(s_addr, c_addr); //--MODIFIED

if (conn_socket < 0)
    goto error;

uint8_t buffer[MAX_PACKET+1];

//ssize_t n = read(conn_socket, buffer, MAX_PACKET);
uint16_t n = read(fd_input, buffer, MAX_PACKET); //--MODIFIED

HTTP_response *response = parse_packet(buffer, n);
if(!response)
    goto error;

//if(!send_response(conn_socket, response))
if(!send_response(STDOUT_FILENO, response)) //--MODIFIED
    goto error;
```



```
int main(int argc, char *argv[]){

    //------------------- MODIFIED -------------------
    if (argc>1)
        fd_input = open(argv[argc-1] , O_RDONLY );
    if(fd_input < 1){
        fprintf(stderr, "Error accessing input file\n");
        exit(-1);
    }
    argc--;
    //------------------------------------------------
```

Aún no has tenido éxito fuzzeando sockets? Ok, echa un vistazo a estos trozos de código

Challenge 2 – Hint 5

- Why is this code linked with **-lz**??

- Por qué esta enlazado el código con **-lz**??

Challenge 2 – My Solution

Challenge 3 - Check your grammar

- I will publish it soon at: https://github.com/antonio-morales/EkoParty_Advanced_Fuzzing_Workshop/
- I will announce Challenge 3 winners next week ☺
- If you have any doubt on it, send me a pm via Twitter @nosoynadiemas

---

- Lo publicaré en breve en: https://github.com/antonio-morales/EkoParty_Advanced_Fuzzing_Workshop/
- Anunciaré los ganadores del Reto 3 la próxima semana ☺
- If you have any doubt on it, send me a pm via Twitter @nosoynadiemas

CONCLUSION

Conclusion

Don’t waste fuzzing iterations. Use your brain first

THE END

THANK YOU!
GRACIAS!

ASK ME ANYTHING

Antonio Morales Maldonado

Twitter: @nosoynadiemas
Email: antoniomoralesmaldonado@gmail.com