12/12/2024

CIBERSEGURIDAD UTFSM

PWN 101

Introducción a la Explotación de Bugs de Corrupción de Memoria

Pablo Aravena - litneet64

CONTENIDOS

Agenda

1. *Promesa*
2. *¿pwn? ¿codigo vulnerable?*
3. *ELF / Process Memory*
4. *Call Conventions*
5. *Tools (Debugging y Dissasembly)*
6. *Demo*
7. *Mas Tecnicas*
8. *Pwn en el Mercado*

LA PROMESA

- ¿Qué es Pwn?
- Buffer Overflow
  - Why’s and How’s
- Más Técnicas
- ¿Hobby o Trabajo?

¿PWN? ¿CODIGO VULNERABLE?

- Ejemplo simple
- gets

```c
#include <stdio.h>
#include <string.h>

int main() {
    char buf[32];   // holds user input
    char passwd[] = "1337pass"; // hardcoded password (very secure)

    printf("\n Enter the password: \n");
    gets(buf);

    /* password validation */
    if (strcmp(passwd, buf) == 0) {
        printf("succesfully logged in\n");
    } else {
        printf("wrong pass!\n");
    }

    return 0;
}
```

NOT USED
NOWADAYS

¿PWN? ¿CODIGO VULNERABLE?

- gets

```
gets(3)                    Library Functions Manual                    gets(3)

NAME
       gets - get a string from standard input (DEPRECATED)

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <stdio.h>

       [[deprecated]] char *gets(char *s);

DESCRIPTION
       Never use this function.

       gets()  reads  a  line  from stdin into the buffer pointed to by s until either a terminating newline or EOF,
       which it replaces with a null byte ('\0').  No check for buffer overrun is performed (see BUGS below).

RETURN VALUE
       gets() returns s on success, and NULL on error or when end of file occurs while no characters have been read.
       However, given the lack of buffer overrun checking, there can be no guarantees that the  function  will  even
       return.
```

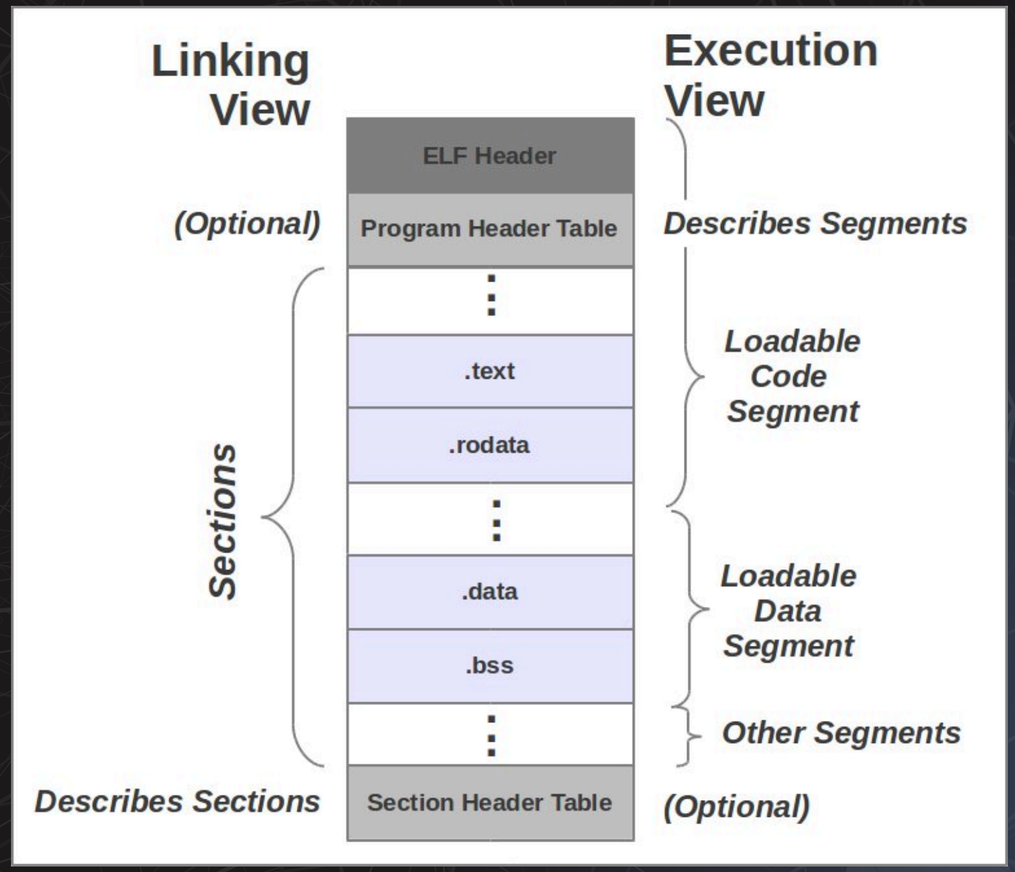
¿PWN? ¿CODIGO VULNERABLE?

- Ejemplo un poco mas elaborado
- BOF no tan obvio

```c
#include <stdio.h>
#include <string.h>

#define BUF_SIZE 64

int main() {
    char c;
    char buf[BUF_SIZE];   // intermediate buffer for dealing with user input
    char userpass[16]; // user password for comparation
    char passwd[] = "1337pass"; // hardcoded password (very secure)

    printf("Enter the password: ");

    for (char i = 0; i < BUF_SIZE; i++) {
        c = getchar();

        if (c == '\n') {
            buf[i] = '\0';
            break;
        }

        buf[i] = c; // very secure buffer write
    }

    strcpy(userpass, buf);

    /* password validation */
    if (strcmp(passwd, userpass) == 0) {
        printf("succesfully logged in\n");
    } else {
        printf("wrong pass!\n");
    }

    return 0;
}
```

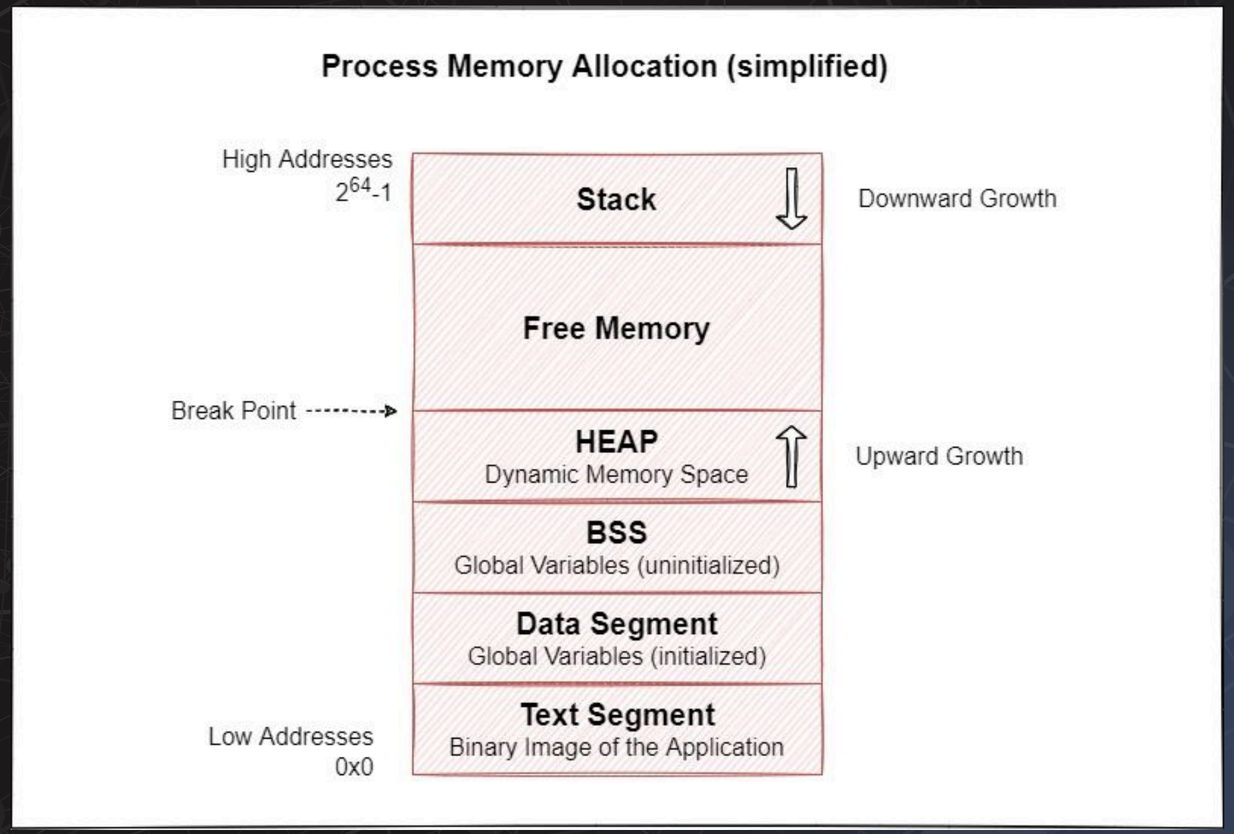
ELF BASICS

- Executable Linux File
- Metadata para Linker
  - ld
  - libc (GLibc, uLibc, varios)



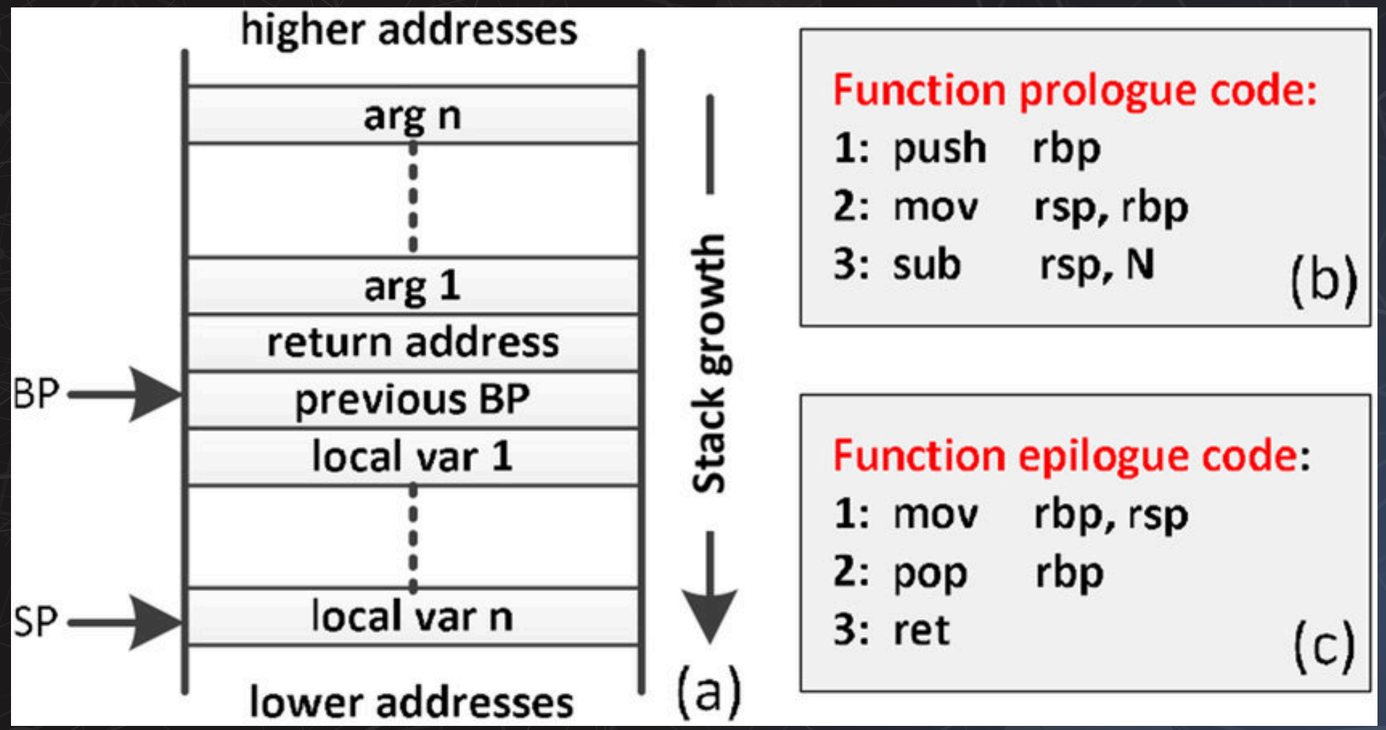
PROCESS MEMORY BASICS

- Programa cargado == Proceso



PROCESS MEMORY BASICS

- Function Call
  - Stack Frame
- x86_32 args
  - Todo por stack
- x86_64
  - rdi
  - rsi
  - rcx
  - rdx
  - r8
  - r9
  - Resto por stack



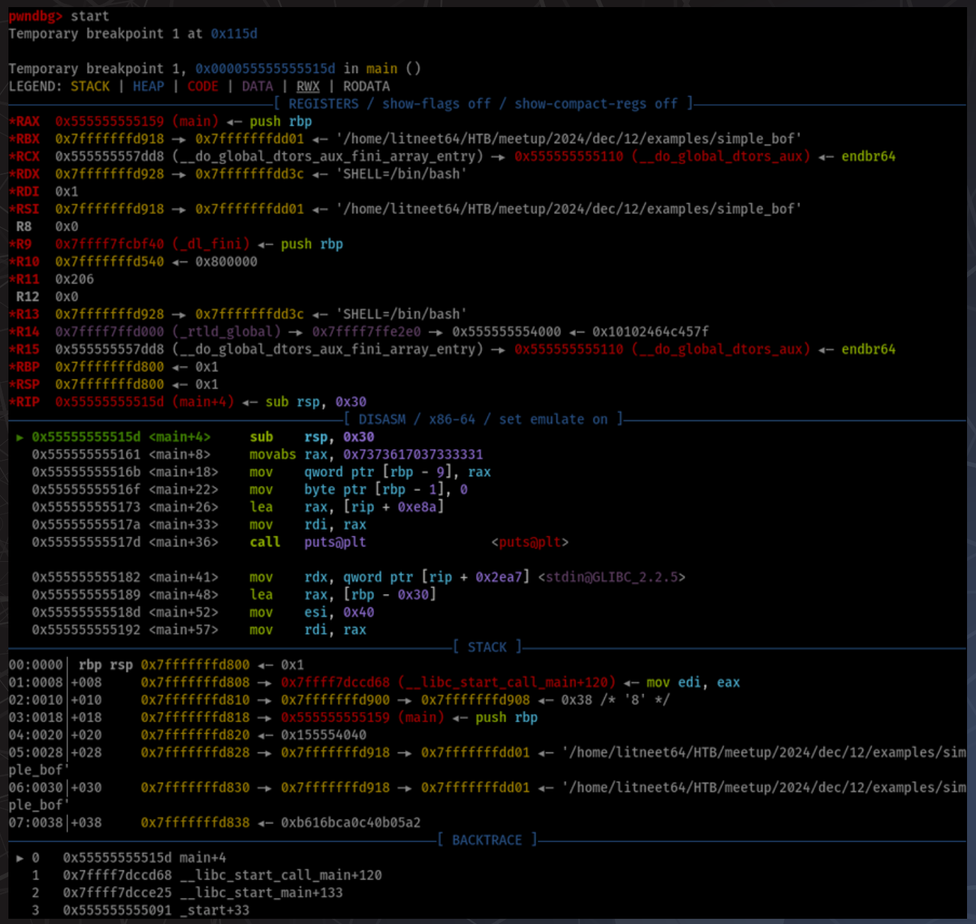
¿PWN? ¿CODIGO VULNERABLE?

- Ejemplo simple
- fgets

```c
#include <stdio.h>
#include <string.h>

int main() {
    char passwd[] = "1337pass"; // hardcoded password (very secure)
    char buf[32];   // holds user input

    printf("Enter the password: \n");

    fgets(buf, 64, stdin);

    /* password validation */
    if (strcmp(passwd, buf) == 0) {
        printf("succesfully logged in\n");
    } else {
        printf("wrong pass!\n");
    }

    return 0;
}
```

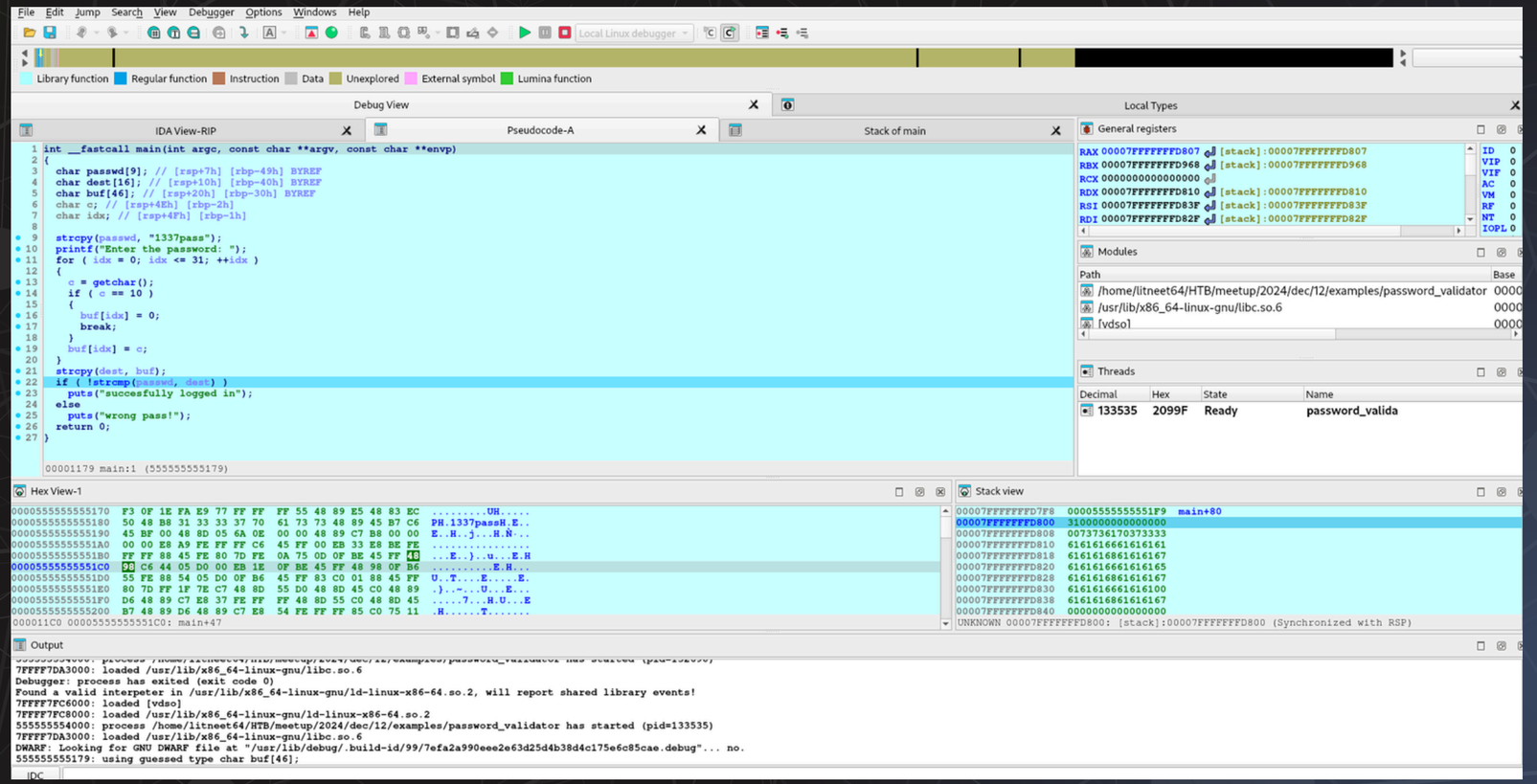
TOOLS - DEBUGGING

- gdb
- pwntools
  - $ pip install pwntools
- pwndbg
  - $ git clone https://github.com/pwndbg/pwndbg
  - $ cd pwndbg
  - $ ./setup.sh
- seer (frontend de gdb)
  - https://github.com/epasveer/seer



TOOLS - DISASSEMBLY

- IDAFree / IDAPro
- Ghidra
- BinaryNinja
- ImHex



HANDS ON!

- Look at my screen!

MAS TECNICAS DE PWN

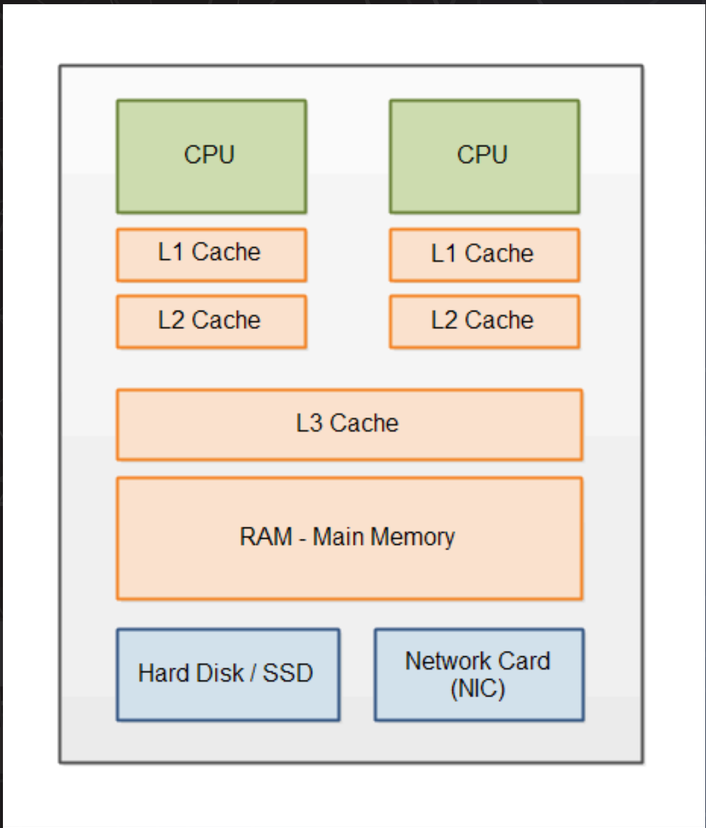
- “Classic” Pwn
  - Executable Shellcode (in RWX stack or mem map)
  - Format string vulnerabilities
  - Canary bypasses through different write-primitives
  - Jump to RSP / “Stack Pivot”
  - Return-Oriented-Programming / ROP-Chaining
  - Leaking Libc address through GOT (ret2libc)
  - Seccomp Filters
  - one_gadget
  - malloc_hooks (older Libc versions)

MAS TECNICAS DE PWN

- Heap-Based and “House of …”
  - tcache-duping (double free, UAF)
  - Leak main arena address with unsorted-bin
  - Read/Write from/to arbitrary places through file descriptor structs (File-Oriented-Programming)
  - “off-by-one” vulns
  - Heap Fengshui

MAS TECNICAS DE PWN

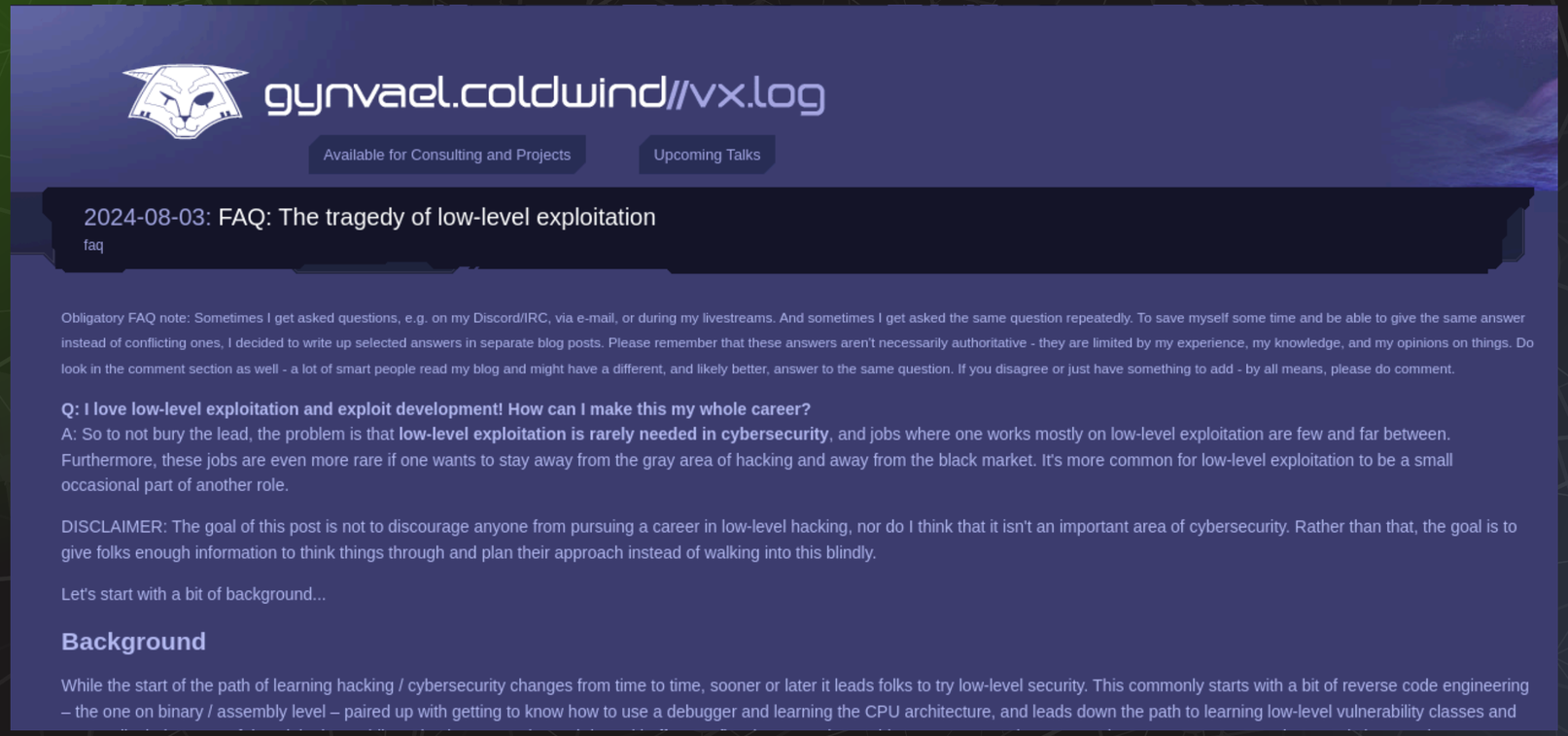
- Microarquitectura de Procesador
  - Cache Timing attack
  - Speculative Execution (Spectre)
  - Address Dereferencing / Prefetching



MAS TECNICAS DE PWN

- Otros
  - Race Conditions (multithreading)
  - Kernel Drivers
  - Kernel vulns (syscalls, slab allocator)
  - Otras arquitecturas
    - ARM (mobile, IoT/embedded)
    - Xtensa (ESP32)
    - MIPS (IoT/embedded)
    - AVR (Arduino)
    - RISC-V (The Future!)
  - Windows
    - userland
    - kernel (where the money is)

“LOW-LEVEL” EN EL MERCADO



gynvael.coldwind//vx.log

Available for Consulting and Projects Upcoming Talks

2024-08-03: FAQ: The tragedy of low-level exploitation

faq

Obligatory FAQ note: Sometimes I get asked questions, e.g. on my Discord/IRC, via e-mail, or during my livestreams. And sometimes I get asked the same question repeatedly. To save myself some time and be able to give the same answer instead of conflicting ones, I decided to write up selected answers in separate blog posts. Please remember that these answers aren't necessarily authoritative - they are limited by my experience, my knowledge, and my opinions on things. Do look in the comment section as well - a lot of smart people read my blog and might have a different, and likely better, answer to the same question. If you disagree or just have something to add - by all means, please do comment.

Q: I love low-level exploitation and exploit development! How can I make this my whole career?
A: So to not bury the lead, the problem is that **low-level exploitation is rarely needed in cybersecurity**, and jobs where one works mostly on low-level exploitation are few and far between. Furthermore, these jobs are even more rare if one wants to stay away from the gray area of hacking and away from the black market. It's more common for low-level exploitation to be a small occasional part of another role.

DISCLAIMER: The goal of this post is not to discourage anyone from pursuing a career in low-level hacking, nor do I think that it isn't an important area of cybersecurity. Rather than that, the goal is to give folks enough information to think things through and plan their approach instead of walking into this blindly.

Let's start with a bit of background...

Background

While the start of the path of learning hacking / cybersecurity changes from time to time, sooner or later it leads folks to try low-level security. This commonly starts with a bit of reverse code engineering – the one on binary / assembly level – paired up with getting to know how to use a debugger and learning the CPU architecture, and leads down the path to learning low-level vulnerability classes and

“LOW-LEVEL” EN EL MERCADO INTERNACIONAL

MEETUP HTB #1

¡MUCHAS GRACIAS POR SU ATENCIÓN!

¿Preguntas?