NYU TANDON SCHOOL OF ENGINEERING

# CSAW CTF ’23 Finals

## NERV Center Walkthrough

```
                               __ _'_','_'__
                         .o8888888888888888P'
                       .d88888888888888888K
         ,8            888888888888888888888boo._
        :88b           888888888888888888888888888b.
         `Y8b          88888888888888888888888888888b.
           `Yb.       d8888888888888888888888888888888b
             `Yb.___.88888888888888888888888888888888888b
               `Y888888888888888888888888888888CG88888P"'
                 `88888888888888888888888888888MM88P"'
"Y888K    "Y8P""Y888888888888888888888888oo._""""
  88888b    8    8888`Y88888888888888888888888oo.
  8"Y8888b  8    8888  ,8888888888888888888888888o,
  8  "Y8888b8    8888""Y8`Y8888888888888888888888b.
  8    "Y8888    8888   Y  `Y8888888888888888888888
  8      "Y88    8888     .d `Y88888888888888888888b
.d8b.      "8  .d8888b..d88P   `Y88888888888888888888
                                 `Y88888888888888888b.
                  "Y888P""Y8b. "Y888888888888888888888
                    888    888   Y888`Y888888888888888
                    888   d88P    Y88b `Y8888888888888
                    888"Y88K"      Y88b dPY8888888888P
                    888  Y88b       Y88dP  `Y88888888b
                    888   Y88b       Y8P     `Y8888888
                  .d888b.  Y88b.      Y        `Y88888
                                                 `Y88K
                                                   `Y8
```

Brendan Dolan-Gavitt

OSIRIS Hack Night, 11/16/2023

CENTER FOR CYBER SECURITY

*The story begins, as many do, on a dark and stormy night…*

*(You all recognize NYU Tandon here, right?)*

*This was before tenure, so I was sitting in my office late at night*

*Watching our PANDA (QEMU-based) malware analysis sandbox*

*(Actually, this is inaccurate—PANDA doesn't put any instrumentation inside the VM)*

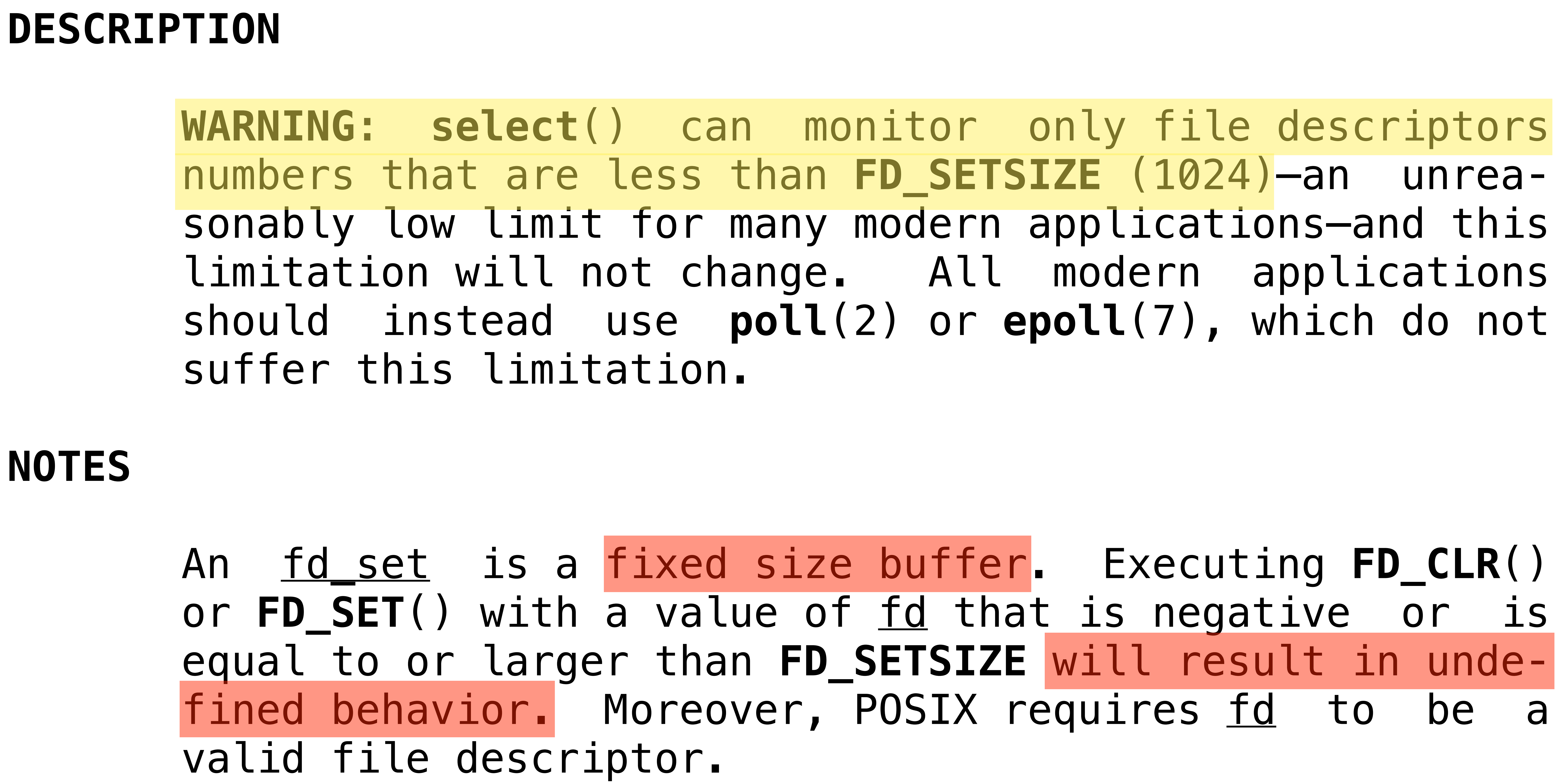
*That’s better—I was looking at the malware sandbox logs*

When I saw something truly frightful!

```
*** buffer overflow detected ***: x86_64-softmmu/qemu-system-x86_64
terminated
======= Backtrace: =========
/lib64/libc.so.6(__fortify_fail+0x37)[0x7f842a2134a7]
/lib64/libc.so.6(+0x35e9d08620)[0x7f842a211620]
/lib64/libc.so.6(+0x35e9d0a417)[0x7f842a213417]
x86_64-softmmu/qemu-system-x86_64(+0x1901fd)[0x7f842f09f1fd]
x86_64-softmmu/qemu-system-x86_64(+0x198388)[0x7f842f0a7388]
x86_64-softmmu/qemu-system-x86_64(main+0xfa9)[0x7f842ef897a9]
/lib64/libc.so.6(__libc_start_main+0xf5)[0x7f842a12aa05]
x86_64-softmmu/qemu-system-x86_64(+0x7ed49)[0x7f842ef8dd49]
```

# What Was the Bug?

## A quick peek at select(2)



**DESCRIPTION**

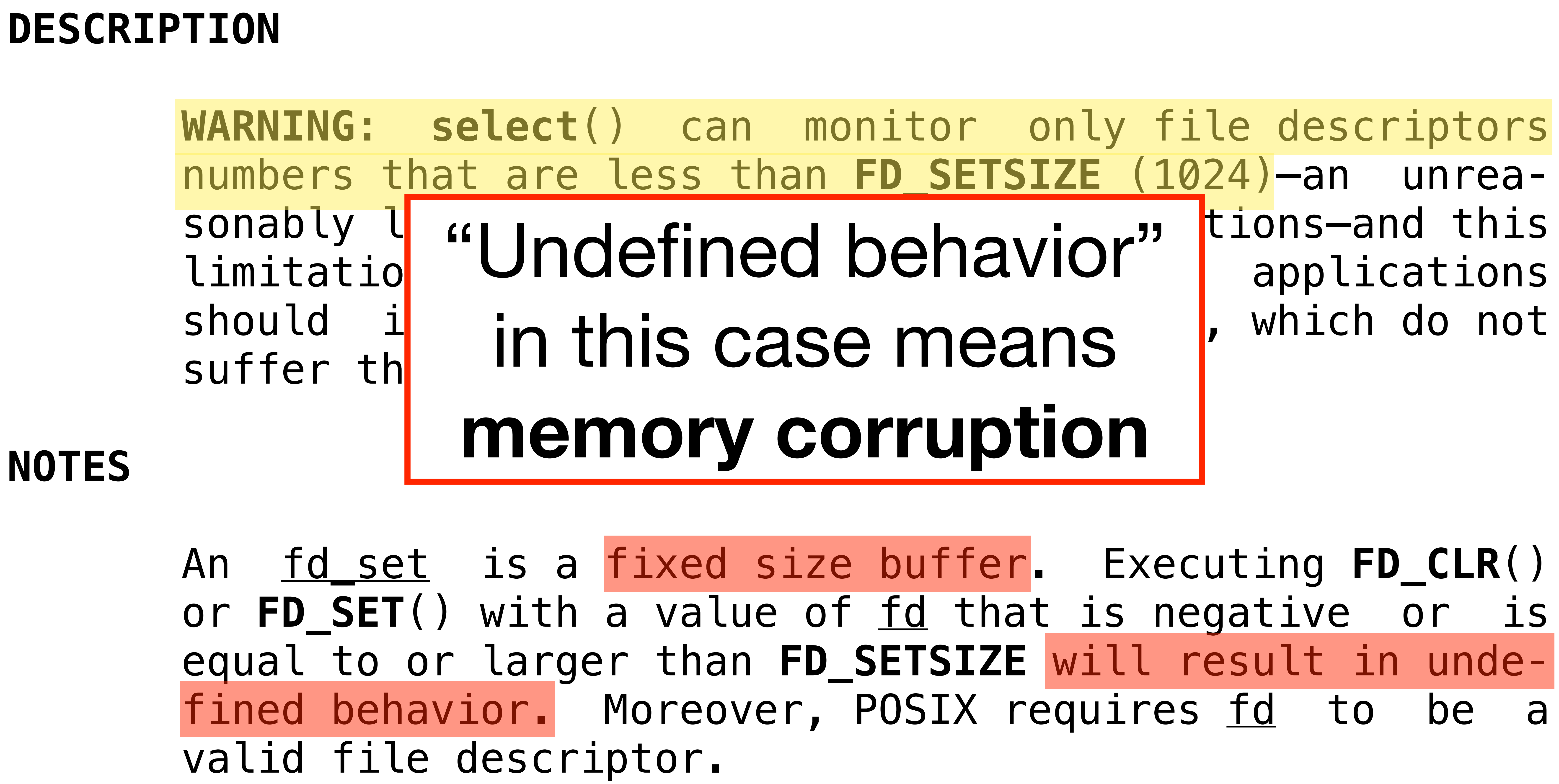
**WARNING: select**() can monitor only file descriptors numbers that are less than **FD_SETSIZE** (1024)—an unreasonably low limit for many modern applications—and this limitation will not change. All modern applications should instead use **poll**(2) or **epoll**(7), which do not suffer this limitation.

**NOTES**

An *fd_set* is a fixed size buffer. Executing **FD_CLR**() or **FD_SET**() with a value of *fd* that is negative or is equal to or larger than **FD_SETSIZE** will result in undefined behavior. Moreover, POSIX requires *fd* to be a valid file descriptor.

# What Was the Bug?

## A quick peek at select(2)



DESCRIPTION

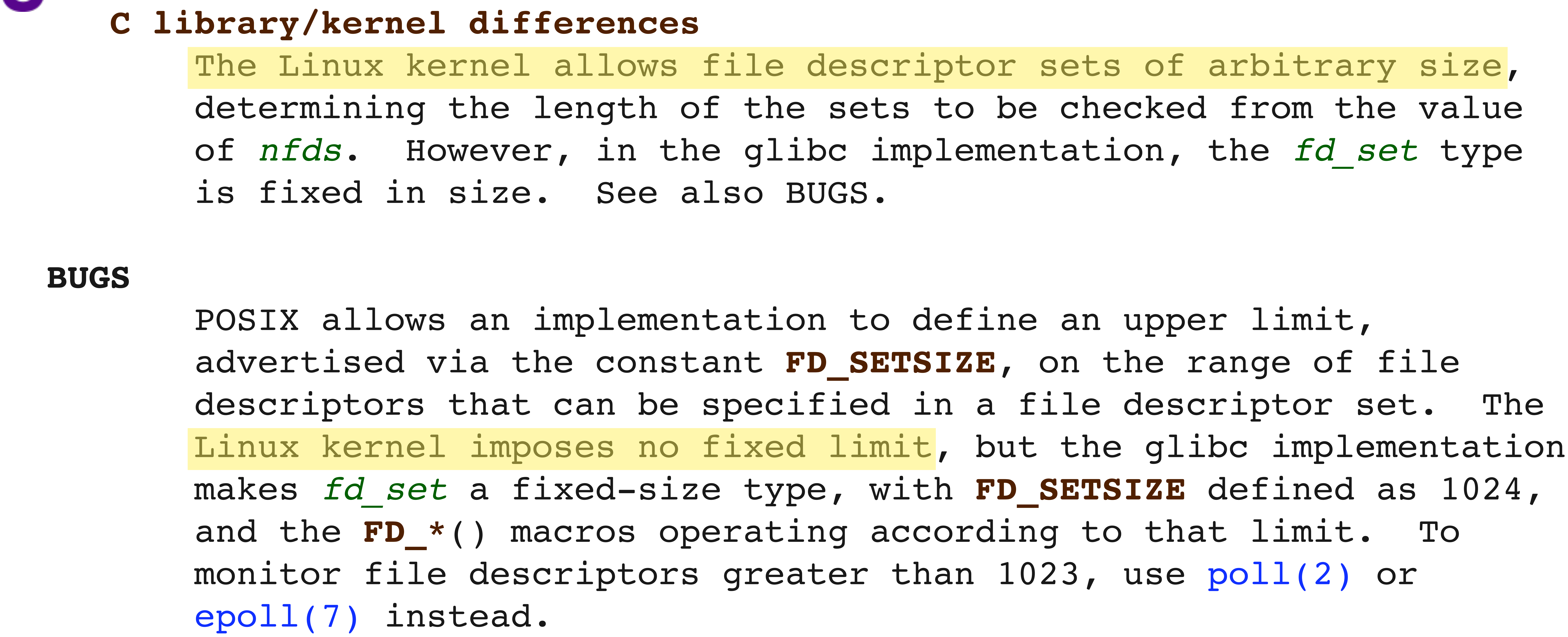
**WARNING: select**() can monitor only file descriptors numbers that are less than **FD_SETSIZE** (1024)—an unrea-
sonably l tions—and this
limitatio applications
should i , which do not
suffer th

NOTES

An fd_set is a fixed size buffer. Executing **FD_CLR**() or **FD_SET**() with a value of fd that is negative or is equal to or larger than **FD_SETSIZE** will result in undefined behavior. Moreover, POSIX requires fd to be a valid file descriptor.

“Undefined behavior” in this case means **memory corruption**

# What About the Kernel?



**C library/kernel differences**

The Linux kernel allows file descriptor sets of arbitrary size, determining the length of the sets to be checked from the value of *nfds*. However, in the glibc implementation, the *fd_set* type is fixed in size. See also BUGS.

**BUGS**

POSIX allows an implementation to define an upper limit, advertised via the constant **FD_SETSIZE**, on the range of file descriptors that can be specified in a file descriptor set. The Linux kernel imposes no fixed limit, but the glibc implementation makes *fd_set* a fixed-size type, with **FD_SETSIZE** defined as 1024, and the **FD_*()** macros operating according to that limit. To monitor file descriptors greater than 1023, use poll(2) or epoll(7) instead.

# In QEMU

## “Probably Overkill”

```c
static fd_set rfds, wfds, xfds;
static int nfds;
static GPollFD poll_fds[1024 * 2]; /* this is probably overkill */
static int n_poll_fds;
static int max_priority;

[...]

static int os_host_main_loop_wait(uint32_t timeout)
{
    struct timeval tv, *tvarg = NULL;
    int ret;

    glib_select_fill(&nfds, &rfds, &wfds, &xfds, &timeout);

    if (timeout < UINT32_MAX) {
        tvarg = &tv;
        tv.tv_sec = timeout / 1000;
        tv.tv_usec = (timeout % 1000) * 1000;
    }

    if (timeout > 0) {
        qemu_mutex_unlock_iothread();
    }

    ret = select(nfds + 1, &rfds, &wfds, &xfds, tvarg);

    if (timeout > 0) {
        qemu_mutex_lock_iothread();
    }

    glib_select_poll(&rfds, &wfds, &xfds, (ret < 0));
    return ret;
}
```

# In QEMU

## “Probably Overkill”

Standard glibc fd_sets, as globals

```
static fd_set rfds, wfds, xfds;
static int nfds;
static GPollFD poll_fds[1024 * 2]; /* this is probably overkill */
static int n_poll_fds;
static int max_priority;

[...]

static int os_host_main_loop_wait(uint32_t timeout)
{
    struct timeval tv, *tvarg = NULL;
    int ret;

    glib_select_fill(&nfds, &rfds, &wfds, &xfds, &timeout);

    if (timeout < UINT32_MAX) {
        tvarg = &tv;
        tv.tv_sec = timeout / 1000;
        tv.tv_usec = (timeout % 1000) * 1000;
    }

    if (timeout > 0) {
        qemu_mutex_unlock_iothread();
    }

    ret = select(nfds + 1, &rfds, &wfds, &xfds, tvarg);

    if (timeout > 0) {
        qemu_mutex_lock_iothread();
    }

    glib_select_poll(&rfds, &wfds, &xfds, (ret < 0));
    return ret;
}
```

# In QEMU

## “Probably Overkill”

No limit on the number of fds

```
static fd_set rfds, wfds, xfds;
static int nfds;
static GPollFD poll_fds[1024 * 2]; /* this is probably overkill */
static int n_poll_fds;
static int max_priority;

[...]

static int os_host_main_loop_wait(uint32_t timeout)
{
    struct timeval tv, *tvarg = NULL;
    int ret;

    glib_select_fill(&nfds, &rfds, &wfds, &xfds, &timeout);

    if (timeout < UINT32_MAX) {
        tvarg = &tv;
        tv.tv_sec = timeout / 1000;
        tv.tv_usec = (timeout % 1000) * 1000;
    }

    if (timeout > 0) {
        qemu_mutex_unlock_iothread();
    }

    ret = select(nfds + 1, &rfds, &wfds, &xfds, tvarg);

    if (timeout > 0) {
        qemu_mutex_lock_iothread();
    }

    glib_select_poll(&rfds, &wfds, &xfds, (ret < 0));
    return ret;
}
```

# In QEMU

## “Probably Overkill”

Passed to select 😩

```
static fd_set rfds, wfds, xfds;
static int nfds;
static GPollFD poll_fds[1024 * 2]; /* this is probably overkill */
static int n_poll_fds;
static int max_priority;

[...]

static int os_host_main_loop_wait(uint32_t timeout)
{
    struct timeval tv, *tvarg = NULL;
    int ret;

    glib_select_fill(&nfds, &rfds, &wfds, &xfds, &timeout);

    if (timeout < UINT32_MAX) {
        tvarg = &tv;
        tv.tv_sec = timeout / 1000;
        tv.tv_usec = (timeout % 1000) * 1000;
    }

    if (timeout > 0) {
        qemu_mutex_unlock_iothread();
    }

    ret = select(nfds + 1, &rfds, &wfds, &xfds, tvarg);

    if (timeout > 0) {
        qemu_mutex_lock_iothread();
    }

    glib_select_poll(&rfds, &wfds, &xfds, (ret < 0));
    return ret;
}
```

# In QEMU

## “Probably Overkill”

**And exposed to guest VM when user mode networking (SLIRP) is enabled 🥴**

```
#ifdef CONFIG_SLIRP
    slirp_update_timeout(&timeout);
    slirp_select_fill(&nfds, &rfds, &wfds, &xfds);
#endif
    qemu_iohandler_fill(&nfds, &rfds, &wfds, &xfds);
    ret = os_host_main_loop_wait(timeout);
    qemu_iohandler_poll(&rfds, &wfds, &xfds, ret);
#ifdef CONFIG_SLIRP
    slirp_select_poll(&rfds, &wfds, &xfds, (ret < 0));
#endif
```

```
static fd_set rfds, wfds, xfds;
static int nfds;
static GPollFD poll_fds[1024 * 2]; /* this is probably overkill */
static int n_poll_fds;
static int max_priority;

[...]

static int os_host_main_loop_wait(uint32_t timeout)
{
    struct timeval tv, *tvarg = NULL;
    int ret;

    glib_select_fill(&nfds, &rfds, &wfds, &xfds, &timeout);

    if (timeout < UINT32_MAX) {
        tvarg = &tv;
        tv.tv_sec = timeout / 1000;
        tv.tv_usec = (timeout % 1000) * 1000;
    }

    if (timeout > 0) {
        qemu_mutex_unlock_iothread();
    }

    ret = select(nfds + 1, &rfds, &wfds, &xfds, tvarg);

    if (timeout > 0) {
        qemu_mutex_lock_iothread();
    }

    glib_select_poll(&rfds, &wfds, &xfds, (ret < 0));
    return ret;
}
```

# What should we do with this?

## Make a CTF challenge obviously

- This bug is interesting for a few reasons:
  - The kernel and glibc have different ideas about the maximum number of fds that can be handled
  - The vuln allows you to set individual bits in the mem corruption
  - The value of those bits is controlled by the status of the file descriptors
    - For example, whether a network connection has any data available for reading

# NERV Center

## In which we weeb out

- I designed a Pwn+Crypto challenge around this core vuln, with a theme based on Neon Genesis Evangelion (1995-1996)
- The main vulnerability is essentially the same: the server opens up a port and uses select() to monitor connections made to it
- The server's ulimit (RLIMIT_NOFILES) is set to 1088 (1024+64), allowing a 64-bit overwrite into the memory after the fd_set

```
                               __ _'_','_'__
                         .o8888888888888888P'
                       .d88888888888888888K
         ,8            888888888888888888888boo._
        :88b           888888888888888888888888888b.
         `Y8b          88888888888888888888888888888b.
           `Yb.       d8888888888888888888888888888888b
             `Yb.___.88888888888888888888888888888888888b
               `Y888888888888888888888888888888CG88888P"'
                 `88888888888888888888888888888MM88P"'
"Y888K    "Y8P""Y888888888888888888888888oo._""""
  88888b    8    8888`Y88888888888888888888888oo.
  8"Y8888b  8    8888  ,8888888888888888888888888o,
  8  "Y8888b8    8888""Y8`Y8888888888888888888888b.
  8    "Y8888    8888   Y  `Y8888888888888888888888
  8      "Y88    8888     .d `Y88888888888888888888b
.d8b.      "8  .d8888b..d88P   `Y88888888888888888888
                                 `Y88888888888888888b.
                  "Y888P""Y8b. "Y888888888888888888888
                    888    888   Y888`Y888888888888888
                    888   d88P    Y88b `Y8888888888888
                    888"Y88K"      Y88b dPY8888888888P
                    888  Y88b       Y88dP  `Y88888888b
                    888   Y88b       Y8P     `Y8888888
                  .d888b.  Y88b.      Y        `Y88888
                                                 `Y88K
                                                   `Y8
```

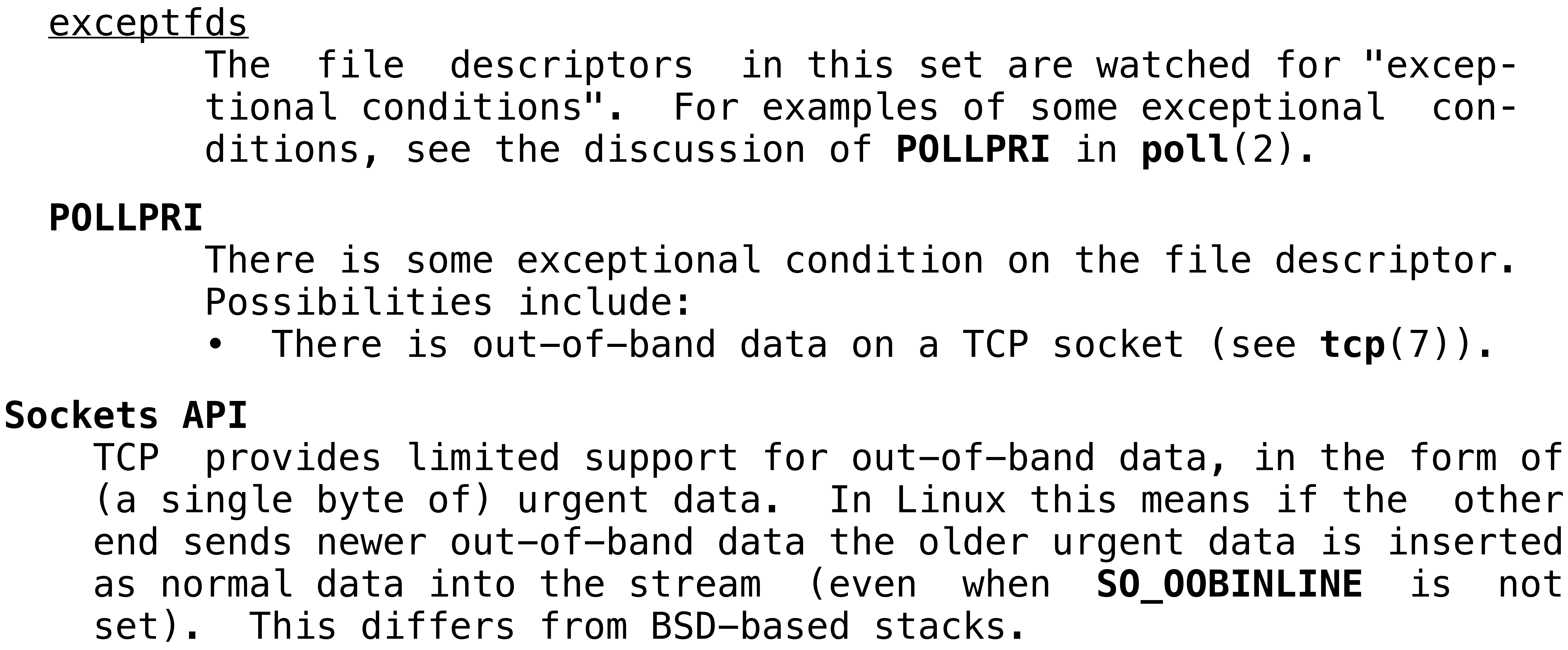
# A Tale of Three fd_sets

## It's *exceptional*

- select() takes three fd_sets to monitor: readfds (fds with data available to read), writefds (fds with data available to write), and exceptfds (???)
- readfds and writefds are a bit hard to control
- The “natural” order to put them in the code means usually the next thing in memory will just be another fd_set, which is not interesting to overwrite
- So what the heck does exceptfds do?

# Out of Band or Out of Bound?

## When you need to send data URGently



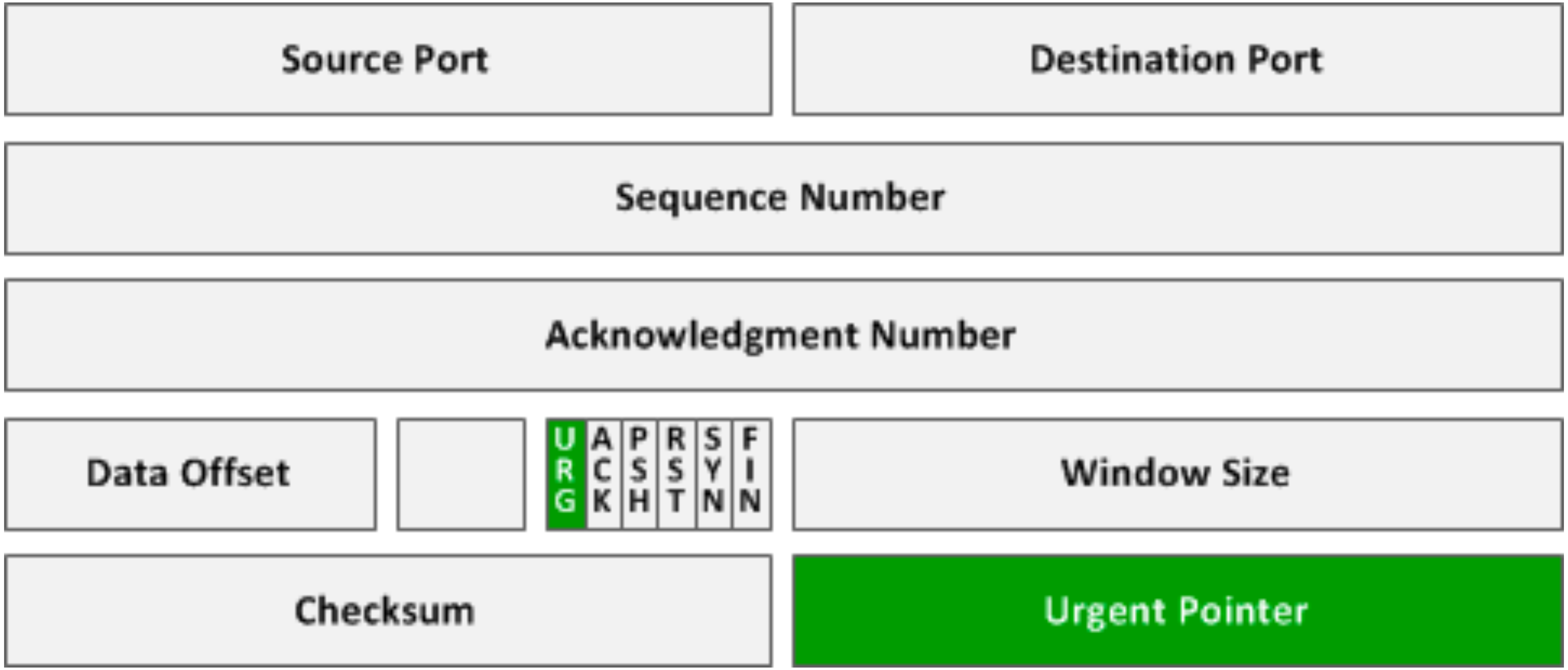
```
   exceptfds
          The  file  descriptors  in this set are watched for "excep-
          tional conditions".  For examples of some exceptional  con-
          ditions, see the discussion of POLLPRI in poll(2).

   POLLPRI
          There is some exceptional condition on the file descriptor.
          Possibilities include:
          •  There is out-of-band data on a TCP socket (see tcp(7)).

Sockets API
       TCP  provides limited support for out-of-band data, in the form of
       (a single byte of) urgent data.  In Linux this means if the  other
       end sends newer out-of-band data the older urgent data is inserted
       as normal data into the stream  (even  when  SO_OOBINLINE  is  not
       set).  This differs from BSD-based stacks.
```

# Out of Band or Out of Bound?

## When you need to send data URGently



Source Port

Destination Port

Sequence Number

Acknowledgment Number

Data Offset

URG ACK PSH RST SYN FIN

Window Size

Checksum

Urgent Pointer

# Out of Band or Out of Bound?

## When you need to send data URGently

- This has some pretty nice properties for a CTF
  - OOB data is pretty obscure and almost never used
  - Until the server actually reads the OOB data, select() will always set that fd bit to 1 – nice and controllable
  - Python lets you easily send OOB data with `sock.send(b'1', socket.MSG_OOB)`

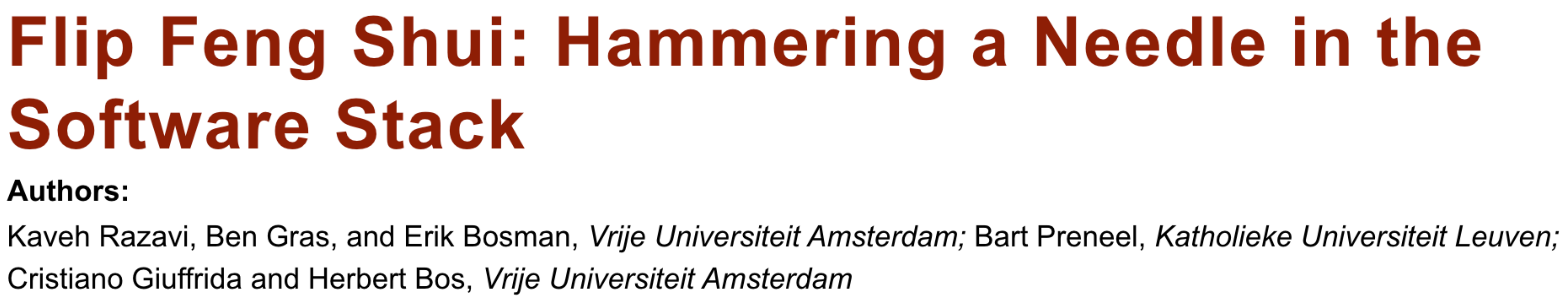
# What Should We Corrupt?

## From Pwn to Pwn+Crypto

- We could just place a function pointer after exceptfds, and have players use that to do a standard pwn and pop a shell
    - But that would be boring
    - Don’t be boring in a CTF challenge
- Instead, let’s kick things up a level and make them solve a crypto problem too
- We’ll put an RSA public modulus (N) into memory right after our exceptfds
- What can you do with control over the first 64 bits of a 1024-bit RSA key?

# Flip Feng Shui

## Getting academic



**Flip Feng Shui: Hammering a Needle in the Software Stack**

**Authors:**

Kaveh Razavi, Ben Gras, and Erik Bosman, *Vrije Universiteit Amsterdam;* Bart Preneel, *Katholieke Universiteit Leuven;* Cristiano Giuffrida and Herbert Bos, *Vrije Universiteit Amsterdam*

- I happened to remember this cool paper from USENIX Security 2016
- The authors wanted to show that RowHammer, which lets you flip random bits in memory, could be used for practical exploitation
- By taking advantage of OS memory deduplication, they could get the memory page they were hammering placed *next to* the page holding the server's ssh RSA public key
- And they showed that when you flip a few bits in an RSA key, *it becomes easy to factor*

# Wait Hold On I Forgot How RSA Works?

## That’s okay I don't do crypto much either

- In RSA we pick two large (e.g. 512-bit) primes **p** and **q**
    - We also pick a public exponent **e**, usually a prime like 65537
- Then the **public key / modulus** is N = p*q
- The secret key **d** is pow(e, -1, N) [the *modular inverse* of e mod N]
    - This is easy to compute if you know p and q, hard if you don't
- We can encrypt / sign a message by doing $m^e$ mod N
- Decrypt with $c^d$ mod N = $(m^e)^d$ mod N = $m^1$ mod N = m

# RSA Authentication

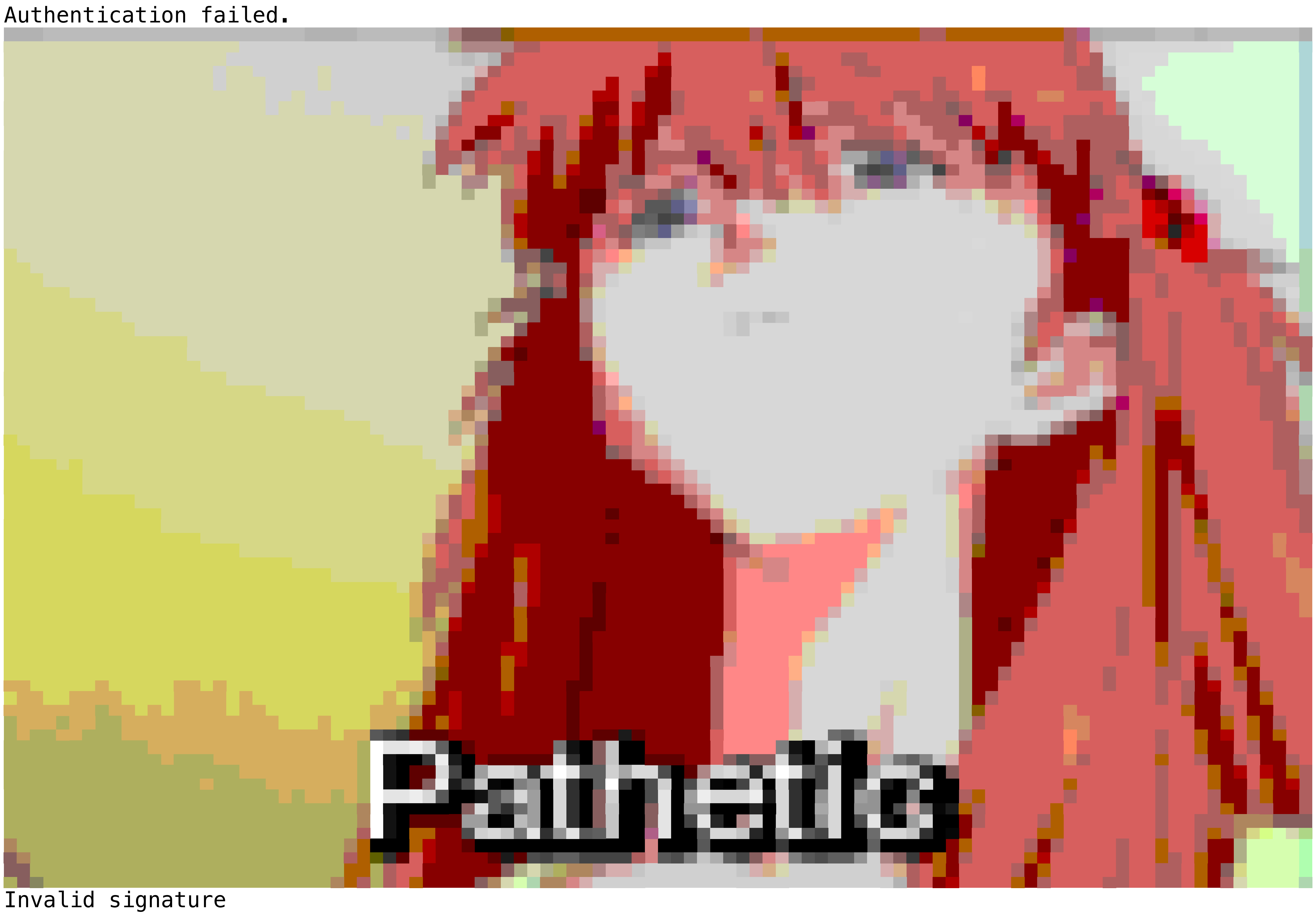
## Challenge-Response in a CTF Challenge

- To use RSA for authentication, the server keeps a public key **N**
- During authentication, it sends a randomly generated *challenge* to the client
- The client uses the private key **d** corresponding to **N** to sign the message and return the signature
- The server then uses **N** to validate the signature
- So if we corrupt **N** , producing **N’**, and factor **N’**, we can forge signatures and the server will accept them as valid with its corrupted key!

# NERV Center Authentication

## [Asuka voice] Pathetic

```
Welcome to the NERV Magi System
Setting up session...
Session sensor port is: 2001
You can connect to this port to view sensor data.
Current authorization level: UNPRIVILEGED
Main menu:
1. Authenticate
2. Print public key
3. Issue sensor system halt
4. Resume sensor operations
5. MAGI status
6. Help
7. Exit
Enter your choice: 1
Challenge: 5ae9dff09cda15bb15db26e76a6668e516fff9201bde283d739bc3469a52fd53
Response: uhhh i don't know
```



```
Authentication failed.
```

```
Invalid signature
```

# An Even More Clever Solution

## That I wish I had thought of

- Stackphish came up with an even more clever solution than just factoring
- Instead of actually factoring the key, you can instead do a search over the 64 bits you control and find a key that makes **N** *prime*
- Then, because of a nice property of Euler's totient function ϕ, we can calculate **d** as

  d = pow(e, -1, N-1)
- Checking primality is fast, and primes are common enough that we're sure to hit one pretty quickly by just picking random values for our 64 bits

# ~Aesthetics~

## In which I get a little carried away

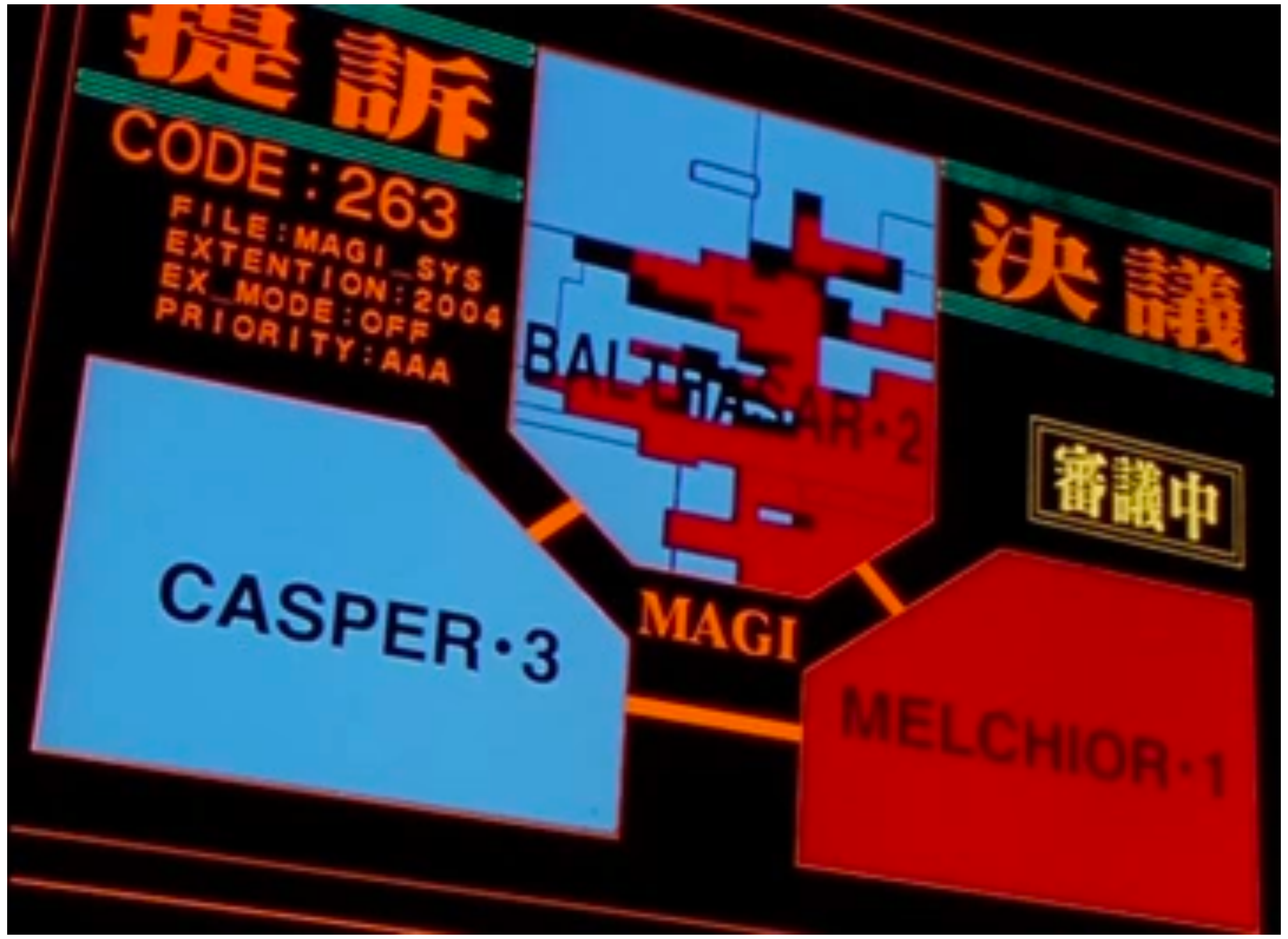
- I wanted to make sure the challenge had good hints, and also looked cool and like something people would want to play with
- I decided to use ANSI colors and unicode characters to add some flavor from the show to the challenge
- Most modern terminals support at least 256 colors and a big chunk of Unicode characters, so you can do some pretty neat things with pure text on the terminal
- You can get pretty elaborate with this (notcurses demo reel): https://www.youtube.com/watch?v=dcjkezf1ARY

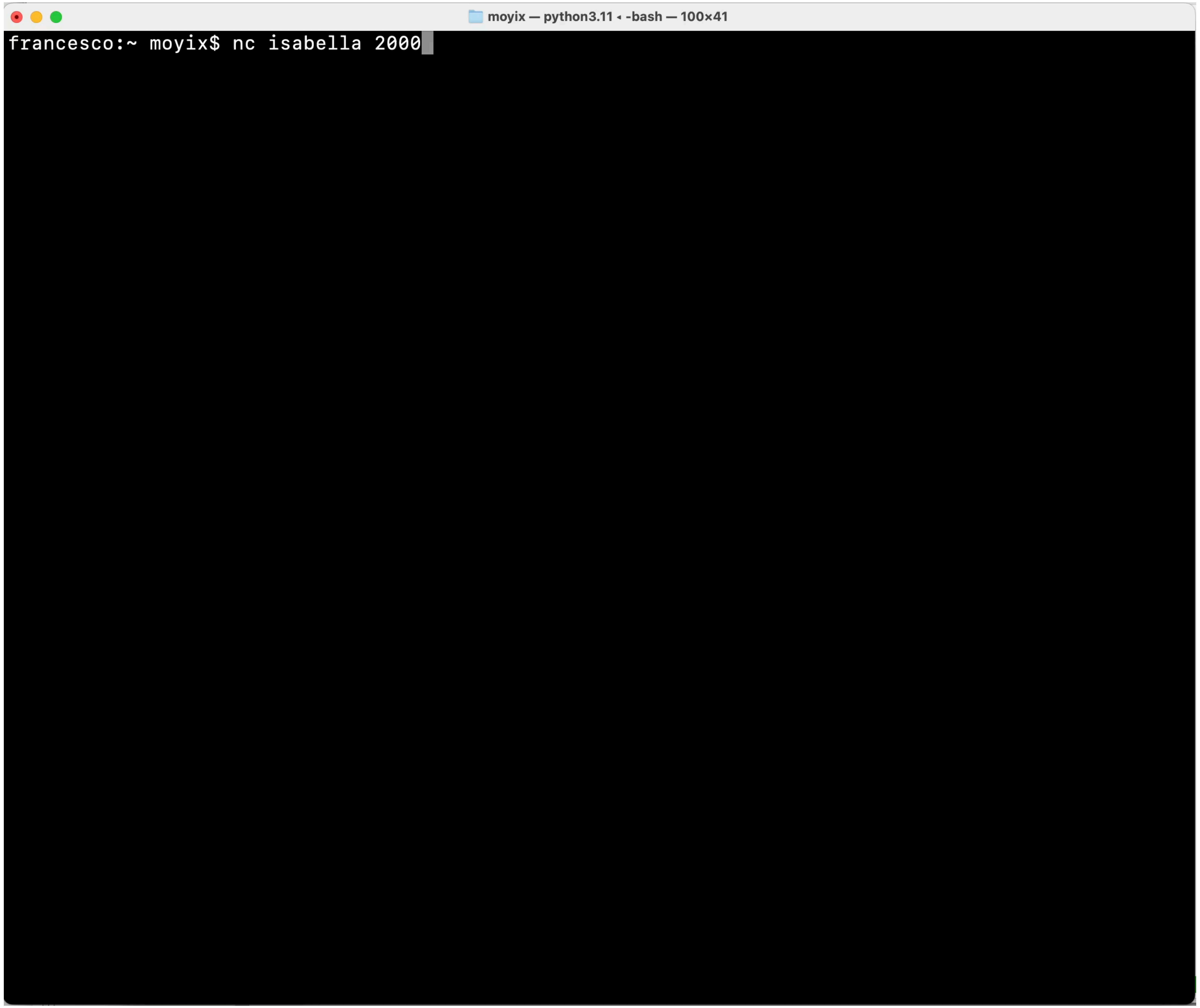
cafe
JC Penn
Two pines mall
1:16 AM

# The MAGI UI

## Three fd_sets, three supercomputers

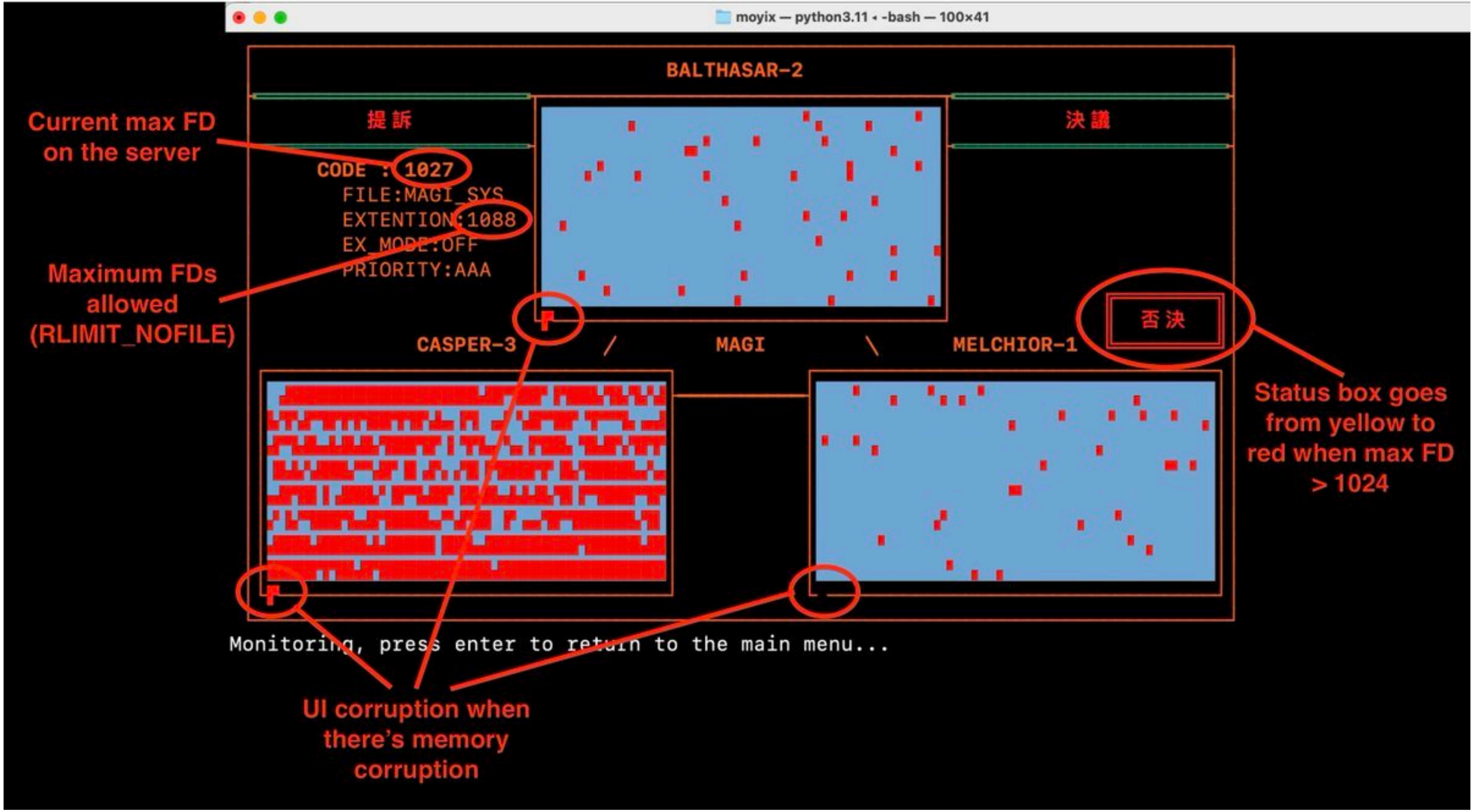
- In the show, the NERV supercomputer consists of three nodes: Casper-Magi 3, Balthasar-Magi 2, and Melchior-Magi 1
- These correspond very nicely to the three fd_sets monitored by select!





```
francesco:~ moyix$ nc isabella 2000
```

# Placing Breadcrumbs in the UI



```
CODE : 1027
  FILE:MAGI_SYS
  EXTENTION:1088
  EX_MODE:OFF
  PRIORITY:AAA
```

```
Monitoring, press enter to return to the main menu...
```

Current max FD on the server

Maximum FDs allowed (RLIMIT_NOFILE)

Status box goes from yellow to red when max FD > 1024

UI corruption when there's memory corruption

# The Credits Easter Egg

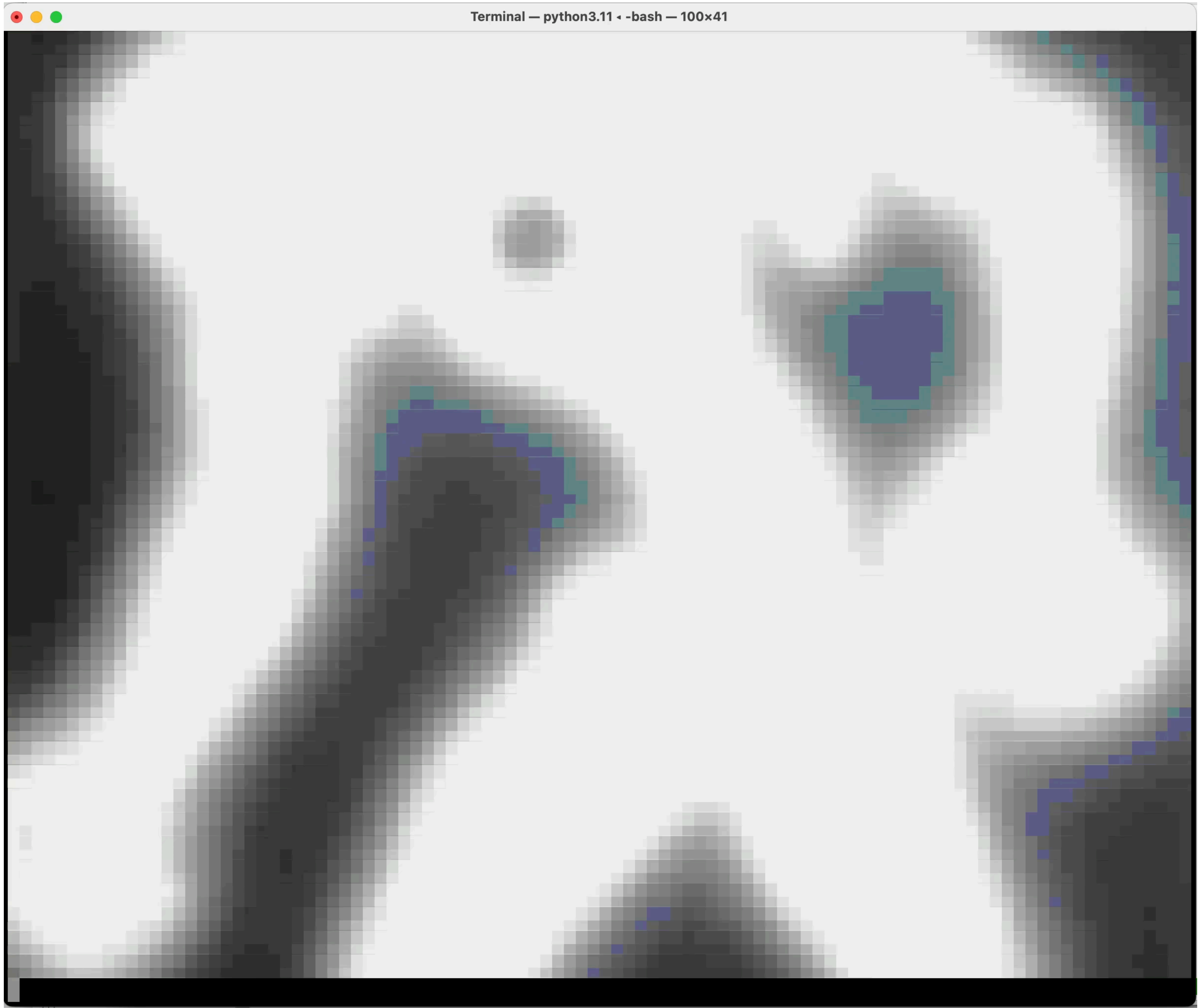
## Even more explicit hints—if you can find them

- During the CTF, we still got no solves until I finally released a hint

  *Look for the easter egg, which has further hints -*
  *what's taking up all that space in the binary?*

- I used the same ANSI+Unicode approach to embed a *full video credit sequence* into the server binary

- You could activate it by connecting to the sensor interface and using the EXAMINE command on three Angels in a row where the first letter of their names spells "RSA" (like Ramiel, Sandalphon, Adam)

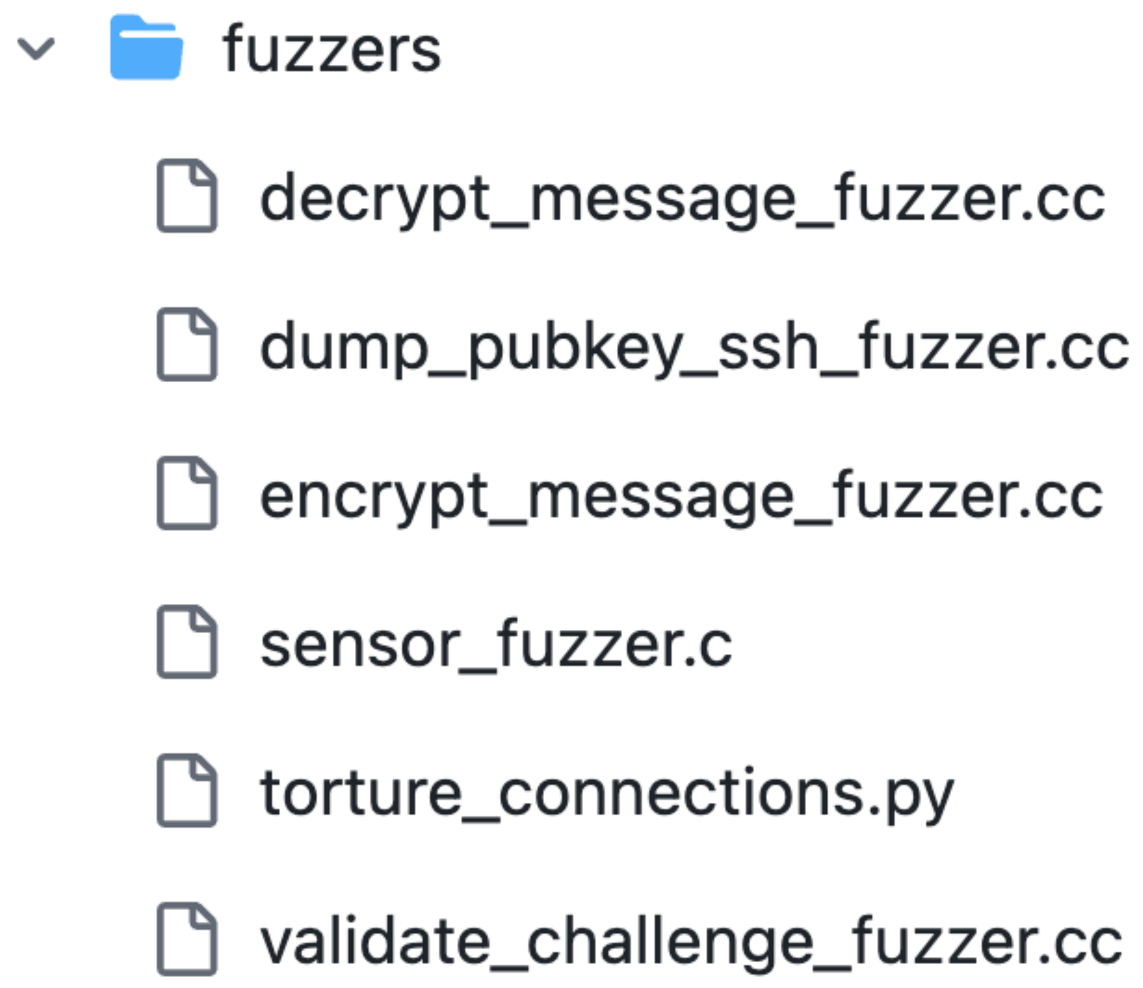


Terminal — python3.11 ◂ -bash — 100×41

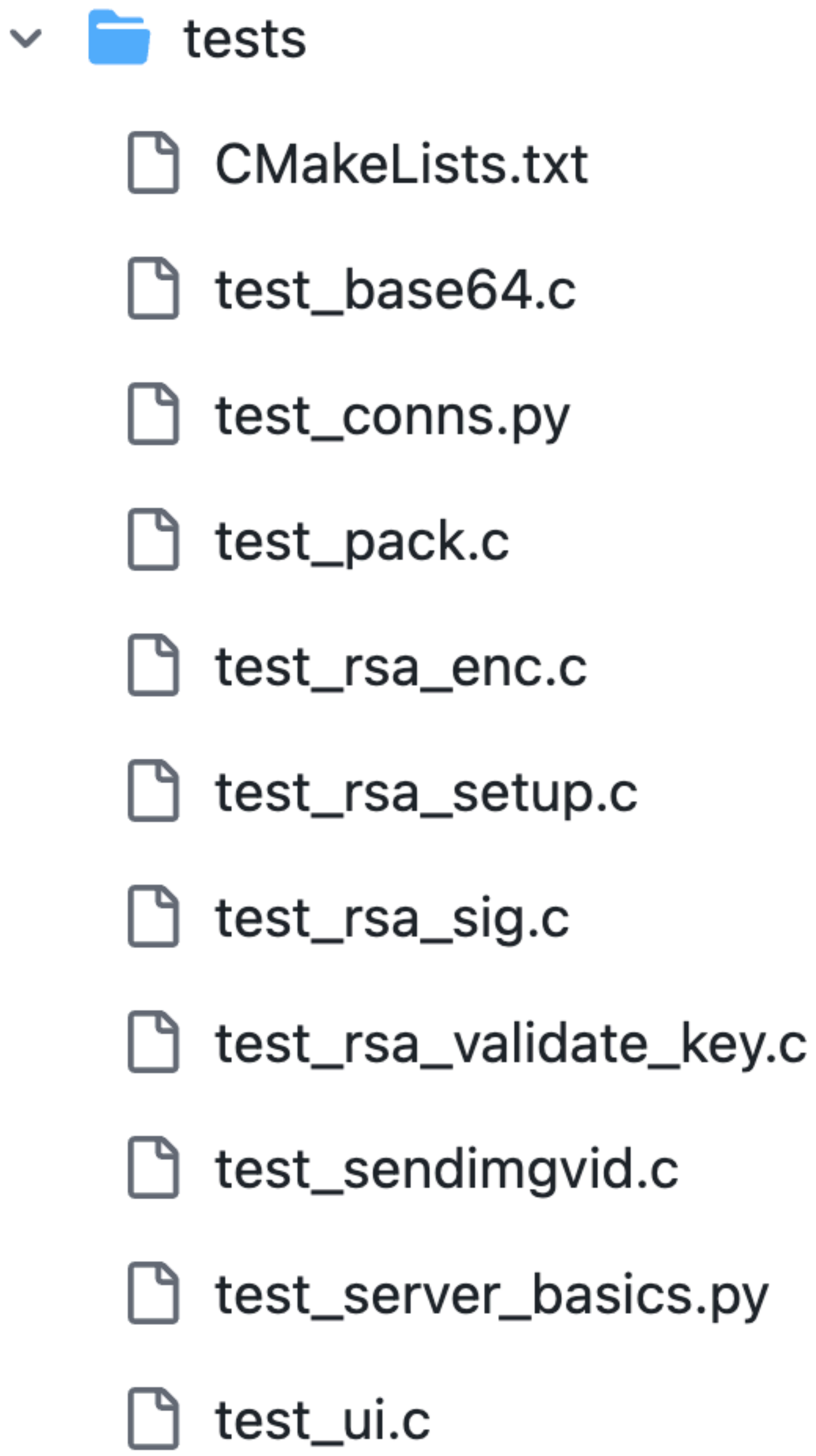
# Behind The Scenes

## The Making Of

- There was also a bunch of extra work that went into making this challenge reliably solvable and avoiding unintentional vulnerabilities!
- I wrote some fuzzers and test cases:



```
fuzzers
    decrypt_message_fuzzer.cc
    dump_pubkey_ssh_fuzzer.cc
    encrypt_message_fuzzer.cc
    sensor_fuzzer.c
    torture_connections.py
    validate_challenge_fuzzer.cc
```

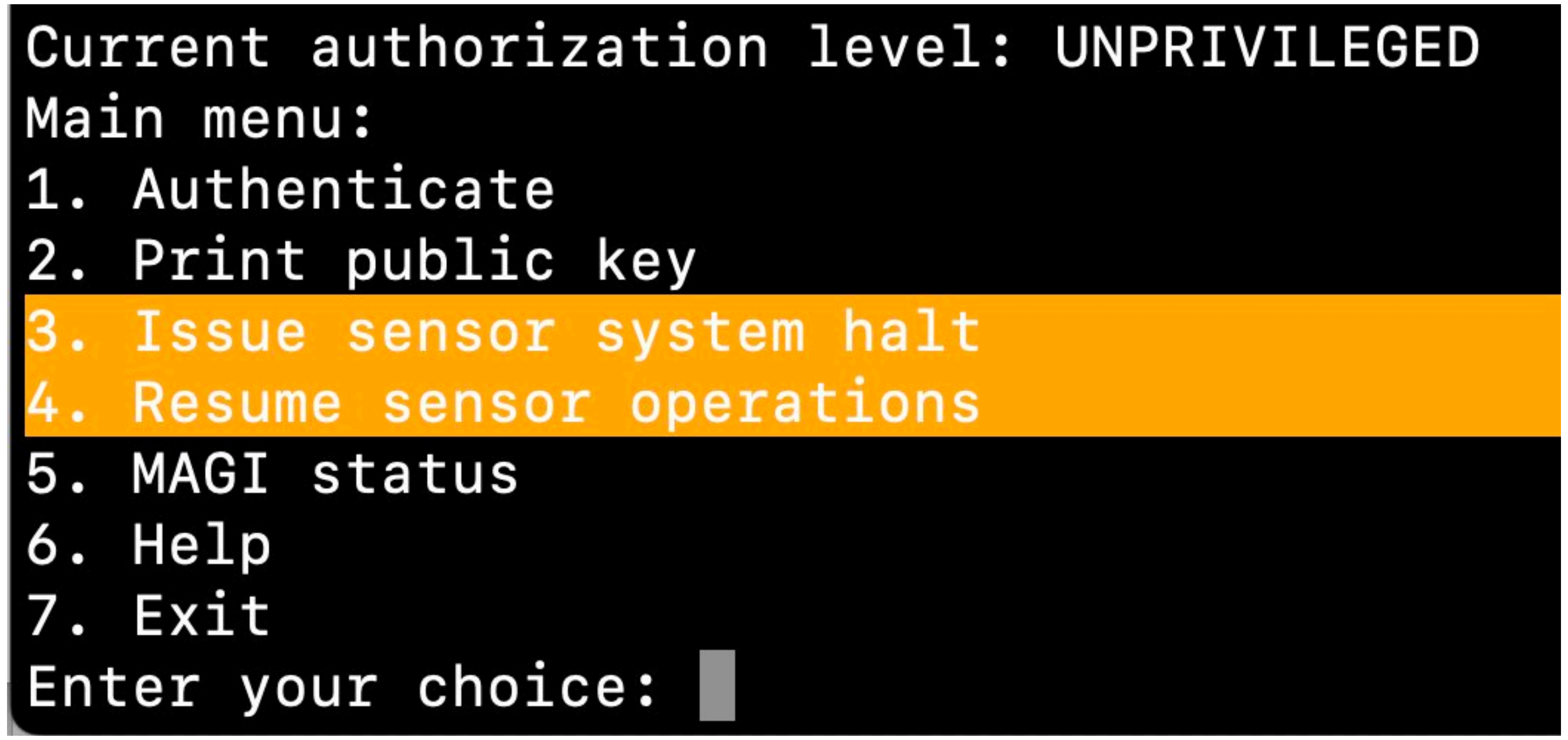


```
tests
    CMakeLists.txt
    test_base64.c
    test_conns.py
    test_pack.c
    test_rsa_enc.c
    test_rsa_setup.c
    test_rsa_sig.c
    test_rsa_validate_key.c
    test_sendimgvid.c
    test_server_basics.py
    test_ui.c
```

# Behind The Scenes

## The Making Of

- select() based vulnerabilities are also annoying because of how select() works
  - You fill up an fd_set with the fds you want to monitor using the FD_SET macro. This sets all those bits to 1 (not attacker controlled).
  - Then you call select(). The kernel updates the fd_sets with the bits corresponding to their actual status (this *is* attacker controlled).
- But this means if you have select() in a loop, half the time you don't control the bits you corrupt!
- I introduced menu options that let you pause the select loop to make it more deterministic



```
Current authorization level: UNPRIVILEGED
Main menu:
1. Authenticate
2. Print public key
3. Issue sensor system halt
4. Resume sensor operations
5. MAGI status
6. Help
7. Exit
Enter your choice:
```

# Making the Credits

## A huge pile of hacks

- To make the credits, I just dumped out all the frames of the opening theme to PNG files
- Then wrote some code that let you provide a subtitle file to overlay text and graphics on each frame, with fade-in/fade-out using transparency
  - …the overlay is done by calling the convert utility from ImageMagick
- Code here, if you dare to read it:

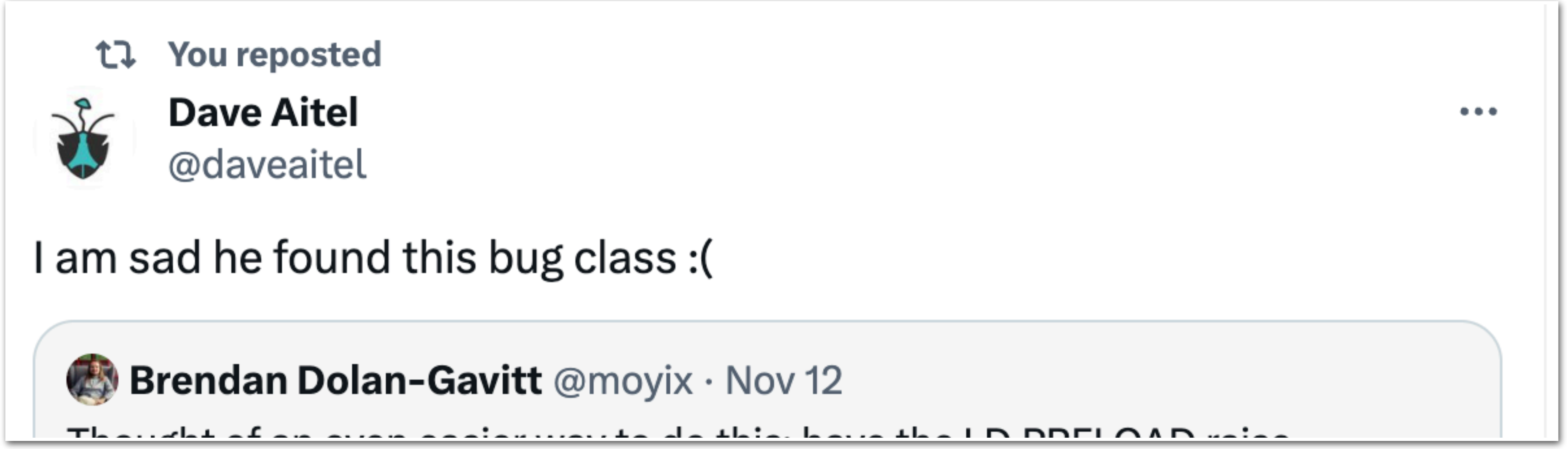
  https://github.com/moyix/csaw23_nervcenter_credits

# Conclusions

## I spent way too much time on this

- This challenge was a huge amount of work
- But also kinda worth it for how much fun people had with it (once they actually started looking at it in earnest)
- Oh, and I may have inadvertently exposed one of Dave Aitel's private bug classes
- Questions?!

https://github.com/moyix/csaw23_nervcenter



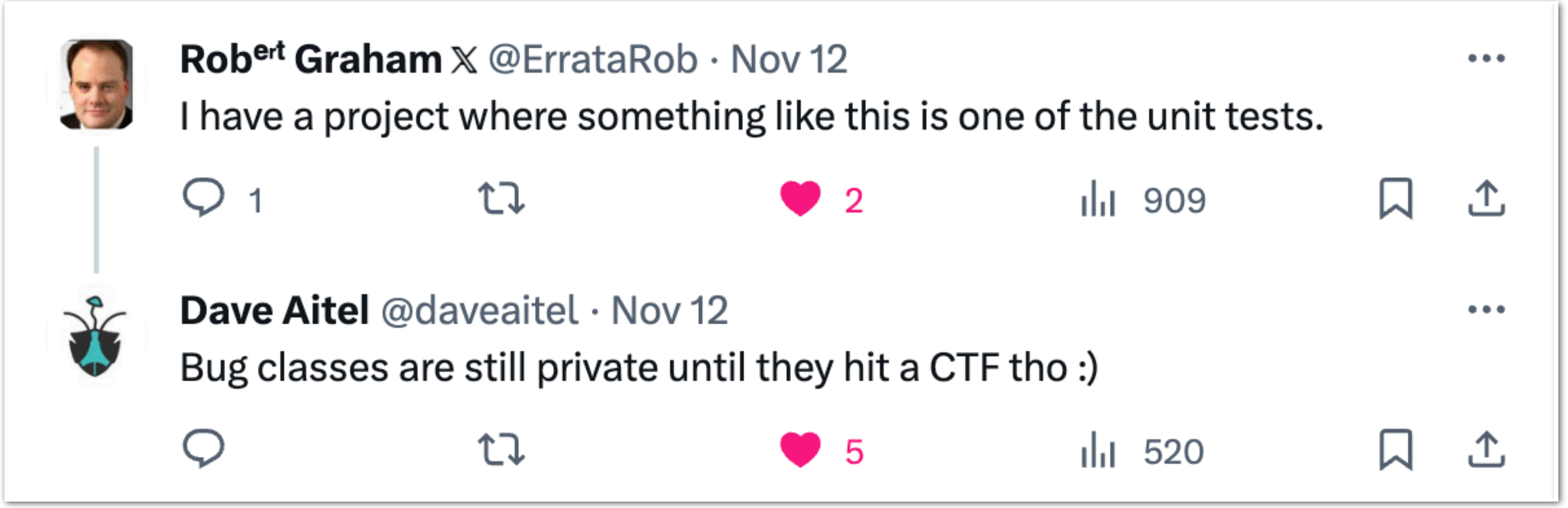
You reposted

Dave Aitel
@daveaitel

I am sad he found this bug class :(

Brendan Dolan-Gavitt @moyix · Nov 12

Thought of an even easier way to do this: have the LD_PRELOAD raise



Robert Graham 𝕏 @ErrataRob · Nov 12

I have a project where something like this is one of the unit tests.

Dave Aitel @daveaitel · Nov 12

Bug classes are still private until they hit a CTF tho :)