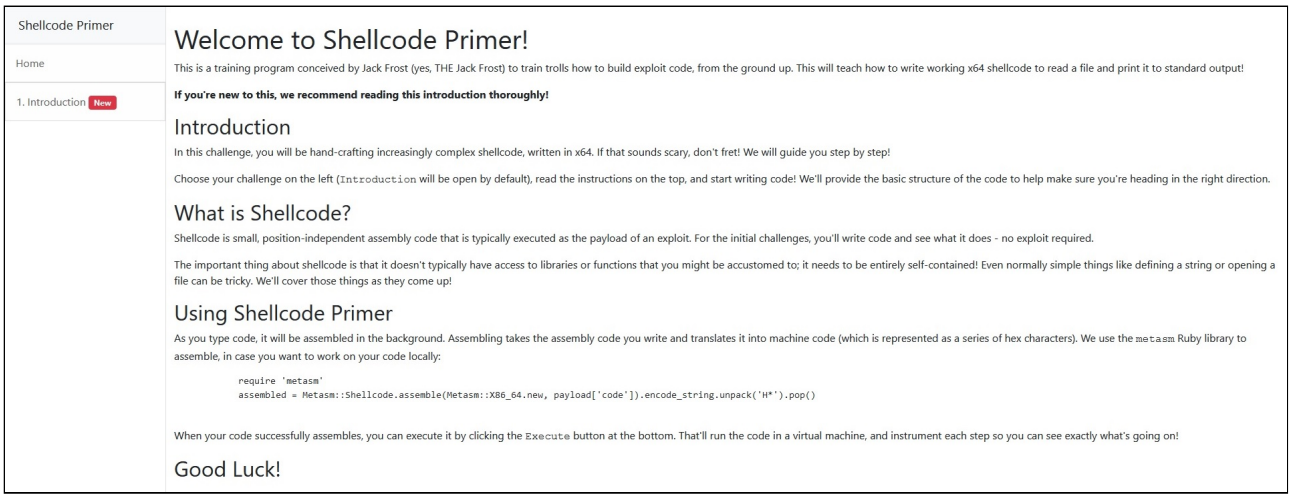
KringleCon 4: Calling Birds!

Jack's Office

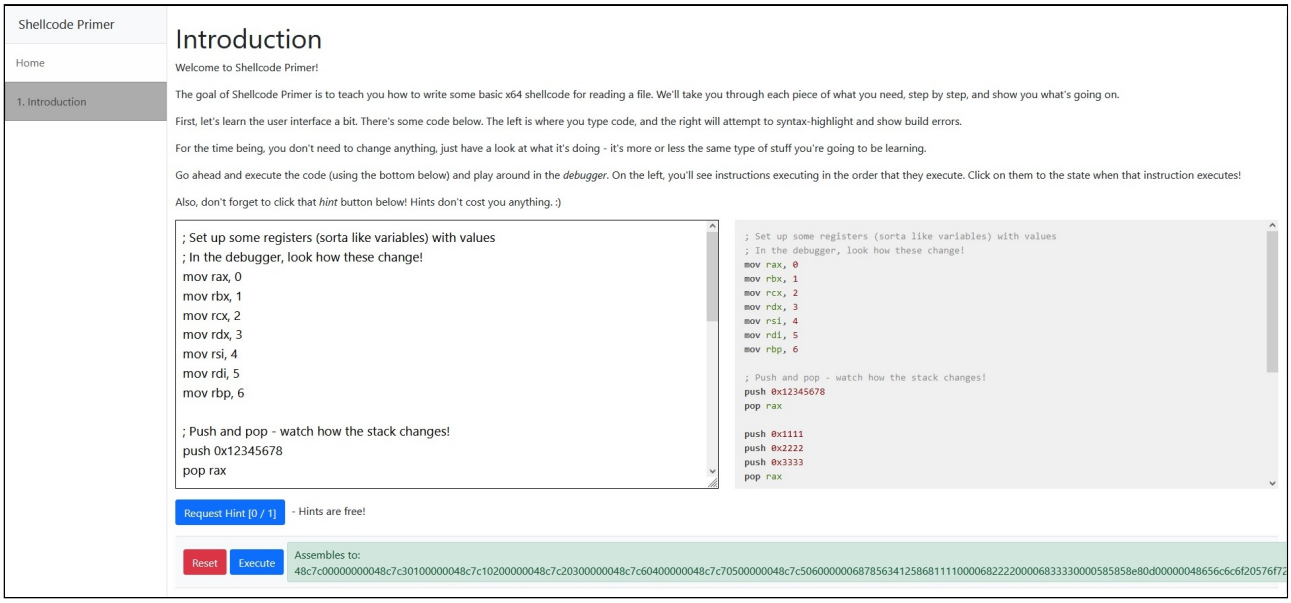
1. Click to talk to `Ruby Cyster`

```
Hey, I'm Ruby Cyster. Don't listen to anything my sister, Ingreta, says about me.

So I'm looking at this system, and it has me a little bit worried.

If I didn't know better, I'd say someone here is learning how to hack North Pole systems.

Who's got that kind of nerve!

Anyway, I hear some elf on the other roof knows a bit about this type of thing.
```

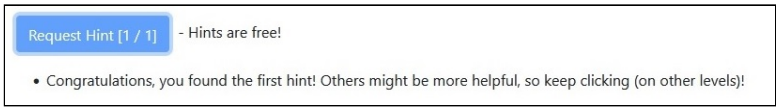
2. Click `Shellcode Primer`



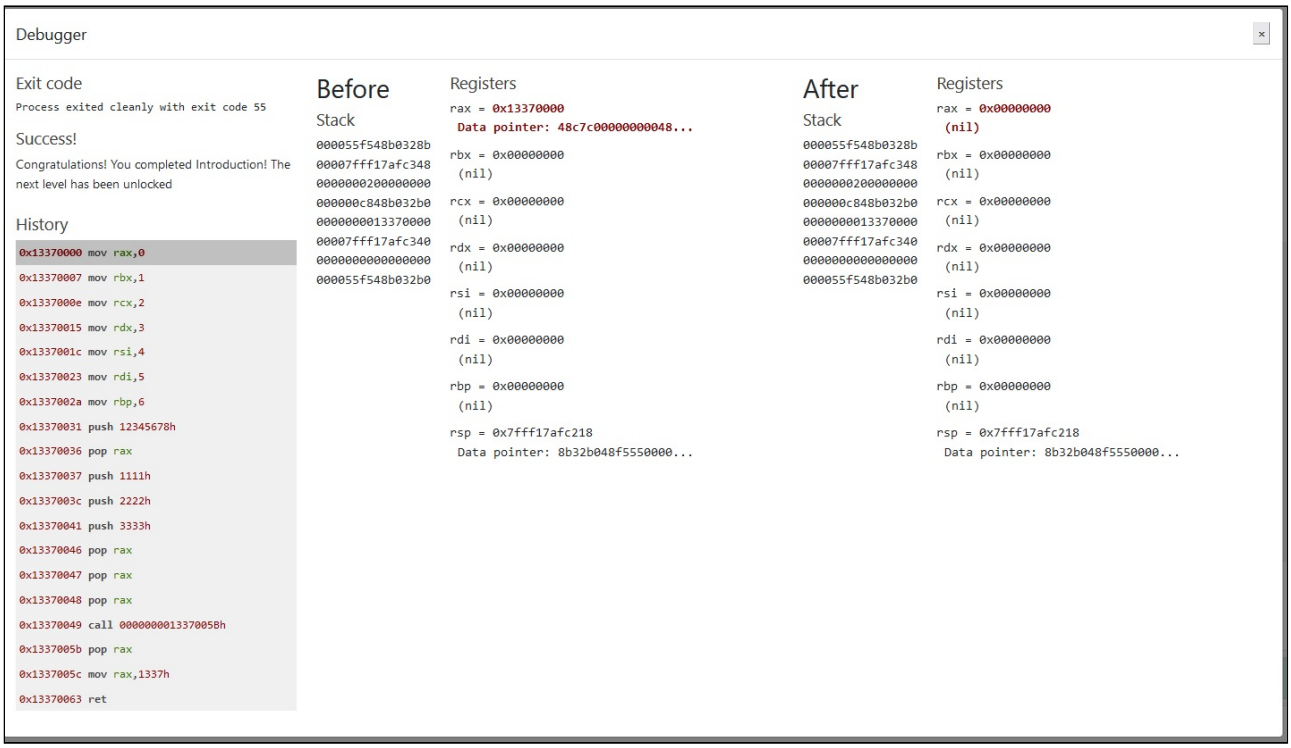
3. Click `1. Introduction`



4. Click the `Request Hint [0/1]` button

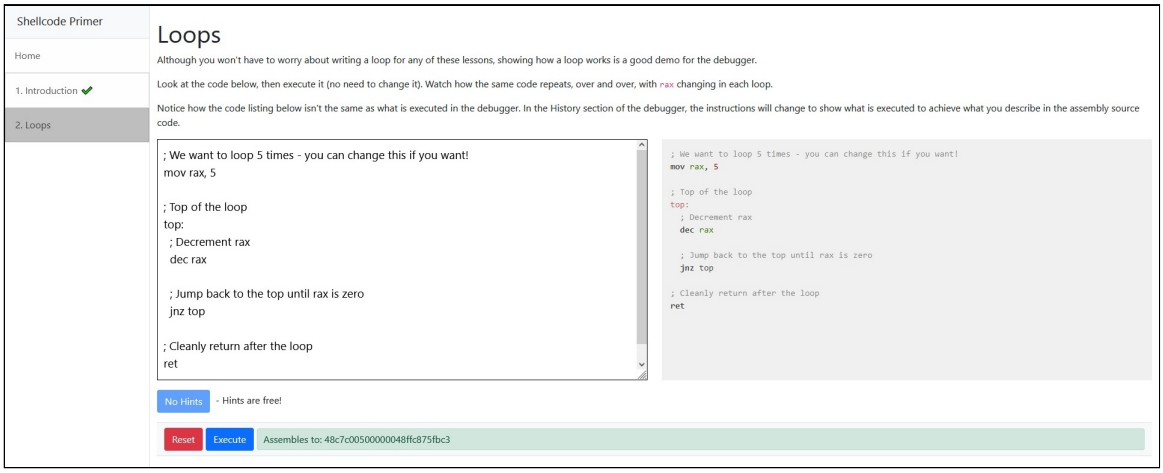


5. Click the `Execute` button



6. Close the Debugger window

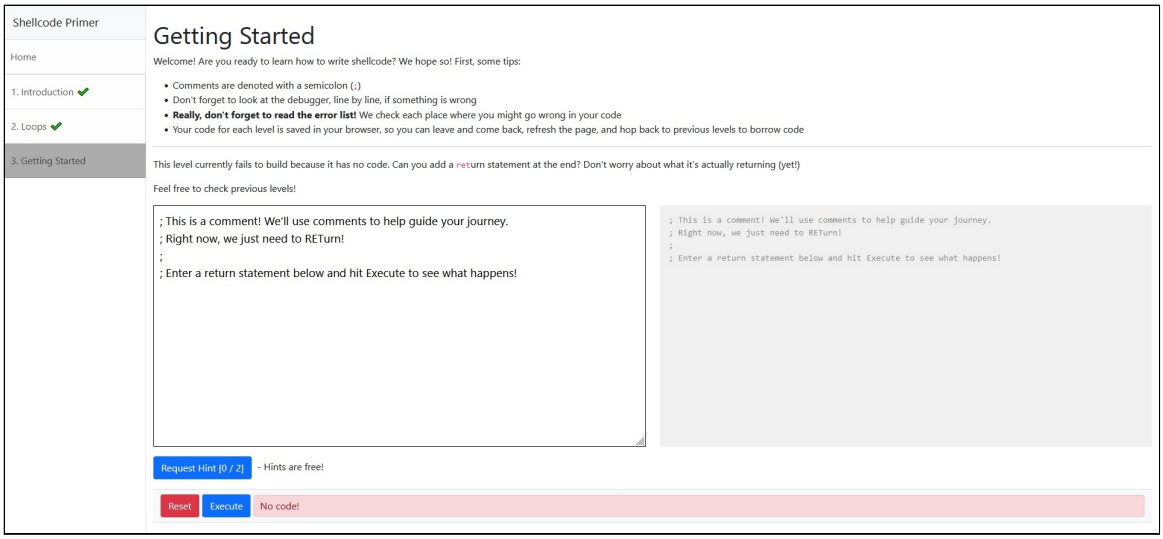
7. Click `2. Loops`



8. Click the `Execute` button

9. Close the Debugger window

10. Click `3. Getting Started`



11. Click the `Request Hint [0/2]` button and then click the `Request Hint [1/2]` button

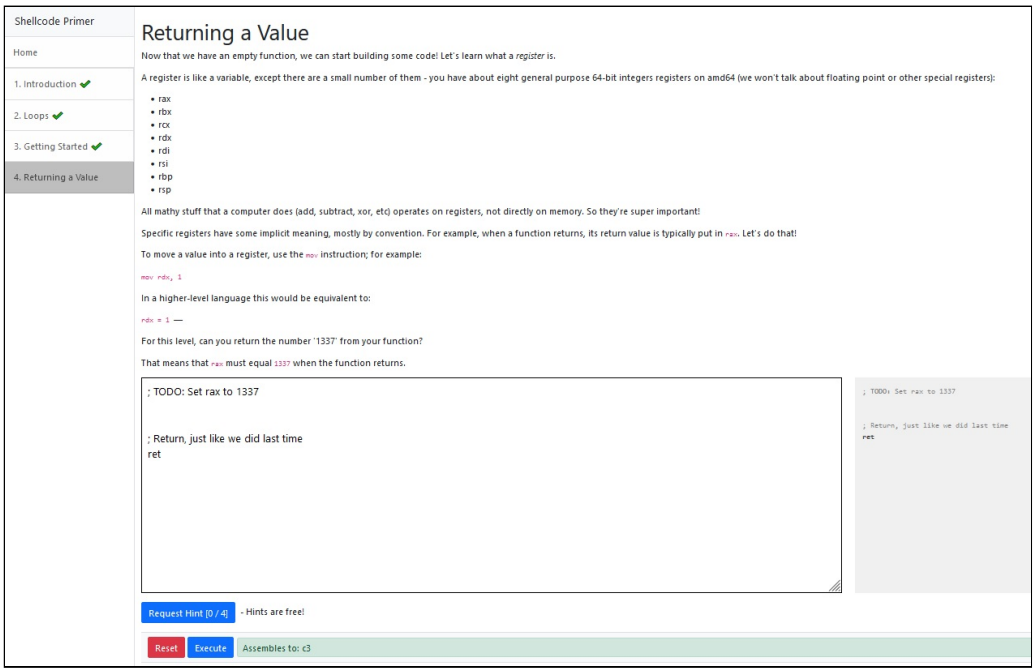
12. Modify the contents of the code window to

```
; This is a comment! We'll use comments to help guide your journey.
; Right now, we just need to RETurn!
;
; Enter a return statement below and hit Execute to see what happens!
ret
```

13. Click the `Execute` button

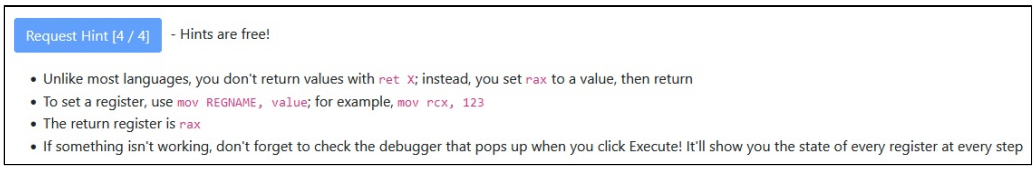
14. Close the Debugger window

15. Click `4. Returning a Value`



16. Click the `Request Hint [0/4]` button and then click the `Request Hint [1/4]` button

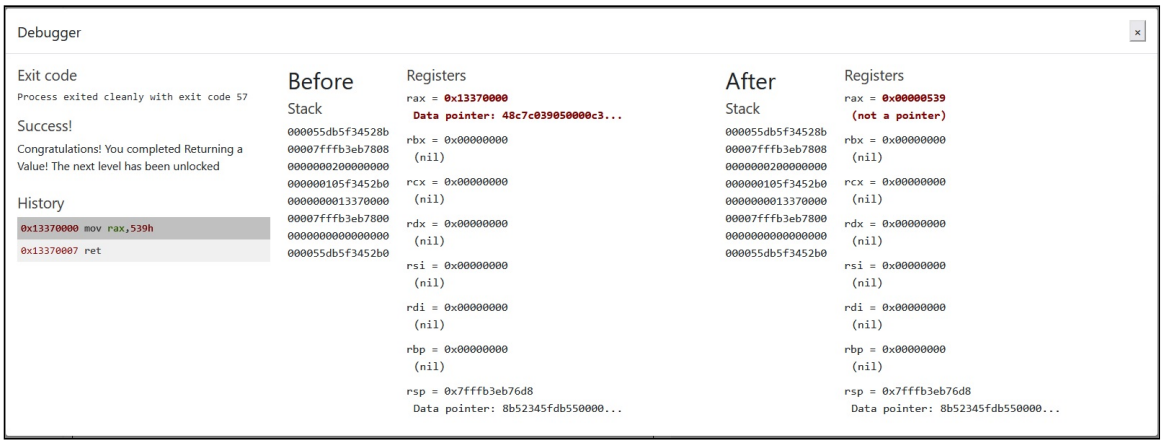
17. Click the `Request Hint [2/4]` button and then click the `Request Hint [3/4]` button



18. Modify the contents of the code window to

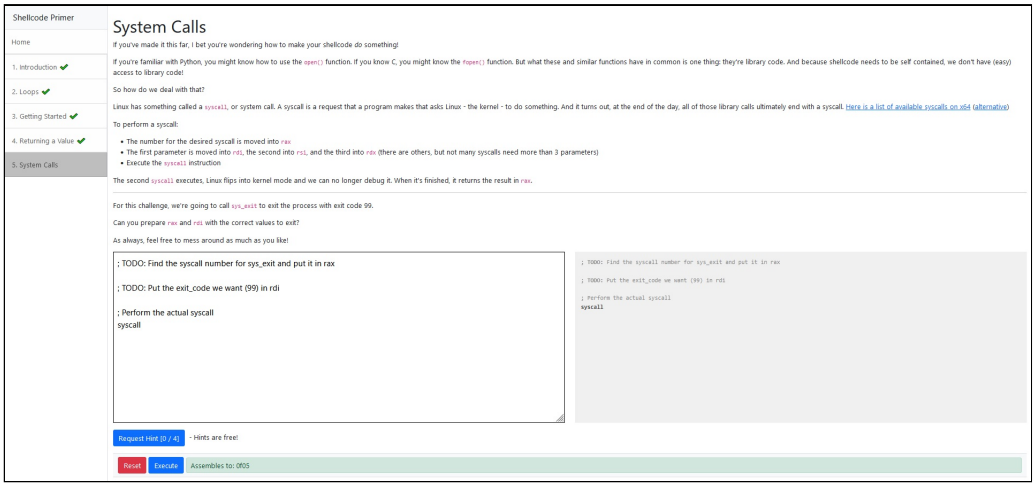
```
; TODO: Set rax to 1337
mov rax, 1337

; Return, just like we did last time
ret
```

19. Click the `Execute` button



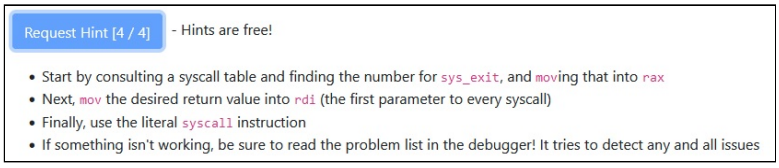
20. Close the Debugger window

21. Click `5. System Calls`



22. Click the `Request Hint [0/4]` button and then click the `Request Hint [1/4]` button

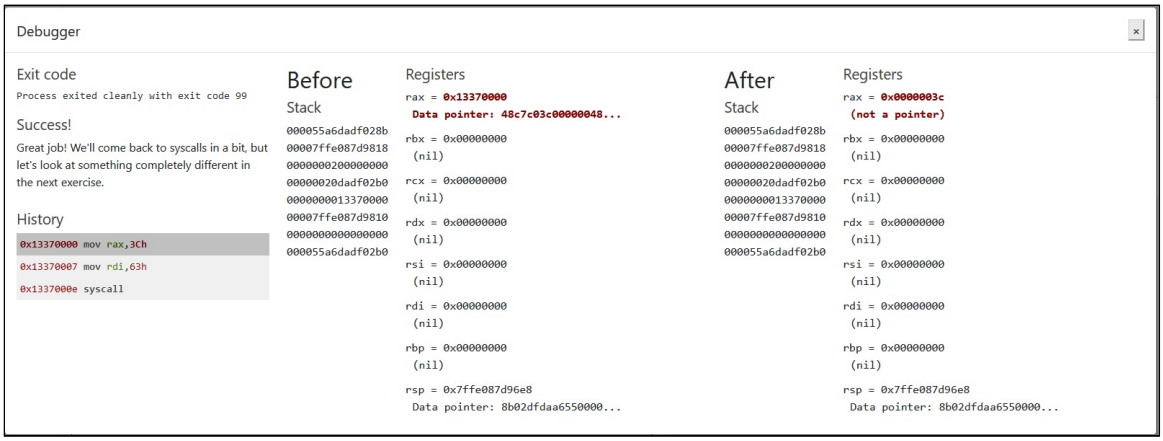
23. Click the `Request Hint [2/4]` button and then click the `Request Hint [3/4]` button



24. Modify the contents of the code window to

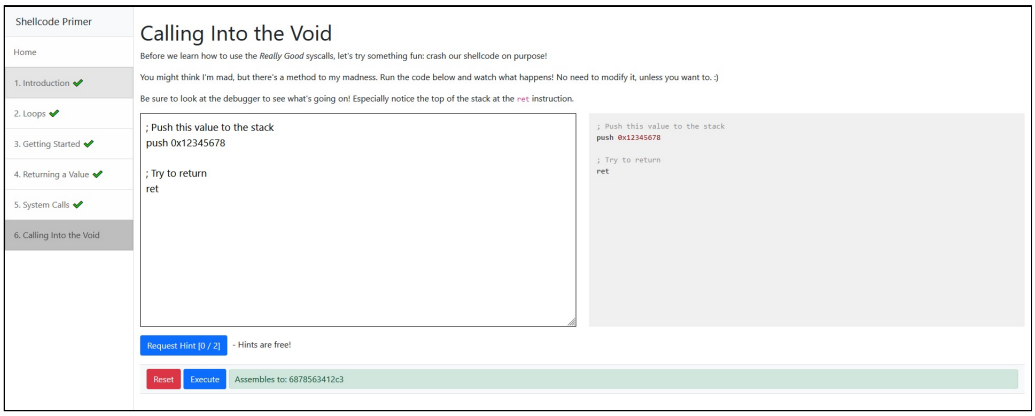
```
; TODO: Find the syscall number for sys_exit and put it in rax
mov rax,  60
; TODO: Put the exit_code we want (99) in rdi
mov rdi, 99
; Perform the actual syscall
syscall
```

25. Click the `Execute` button

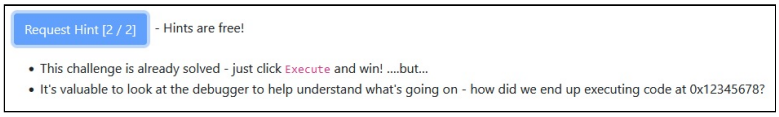


26. Close the Debugger window

27. Click `6. Call Into the Void`



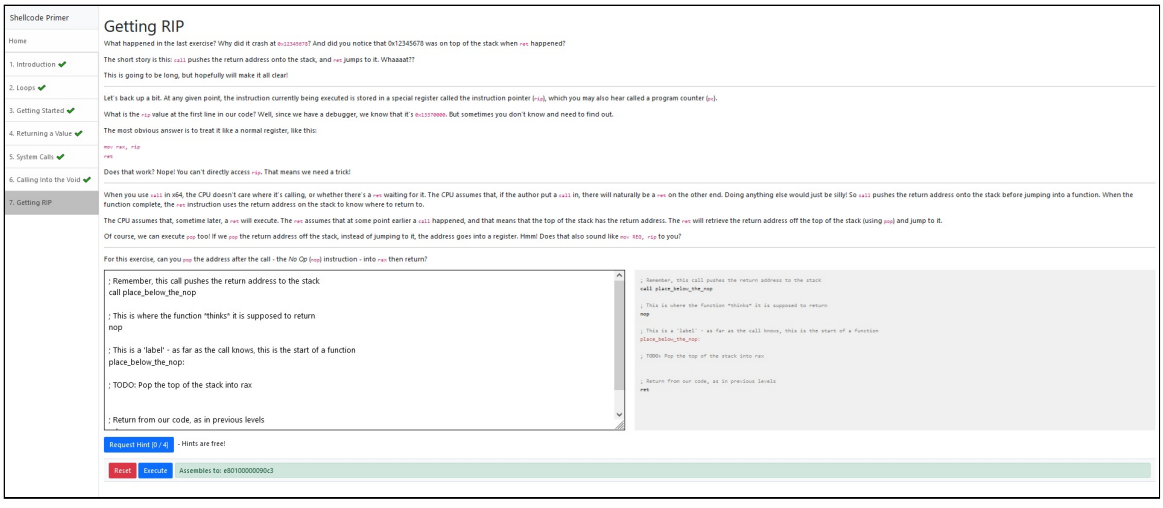
28. Click the `Request Hint [0/2]` button and then click the `Request Hint [1/2]` button



29. Click the `Execute` button

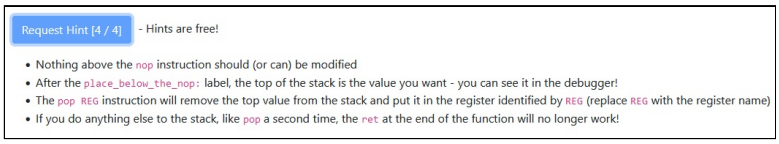
30. Close the Debugger window

31. Click `7. Getting RIP`



32. Click the `Request Hint [0/4]` button and then click the `Request Hint [1/4]` button

33. Click the `Request Hint [2/4]` button and then click the `Request Hint [3/4]` button

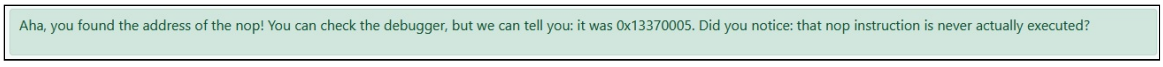


34. Modify the contents of the code window to

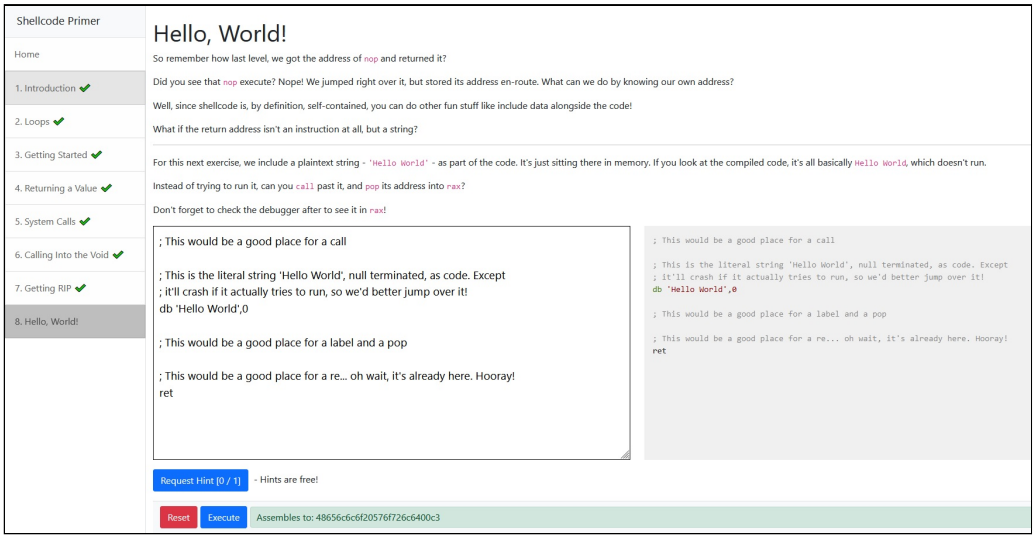
```
; Remember, this call pushes the return address to the stack
call place_below_the_nop

; This is where the function *thinks* it is supposed to return
nop

; This is a 'label' - as far as the call knows, this is the start of a function
place_below_the_nop:

; TODO: Pop the top of the stack into rax
pop rax

; Return from our code, as in previous levels
ret
```

35. Click the `Execute` button

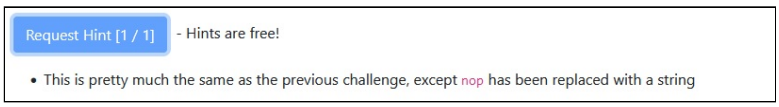
36. Close the Debugger window



37. Click `8. Hello, World!`



38. Click the `Request Hint [0/1]` button



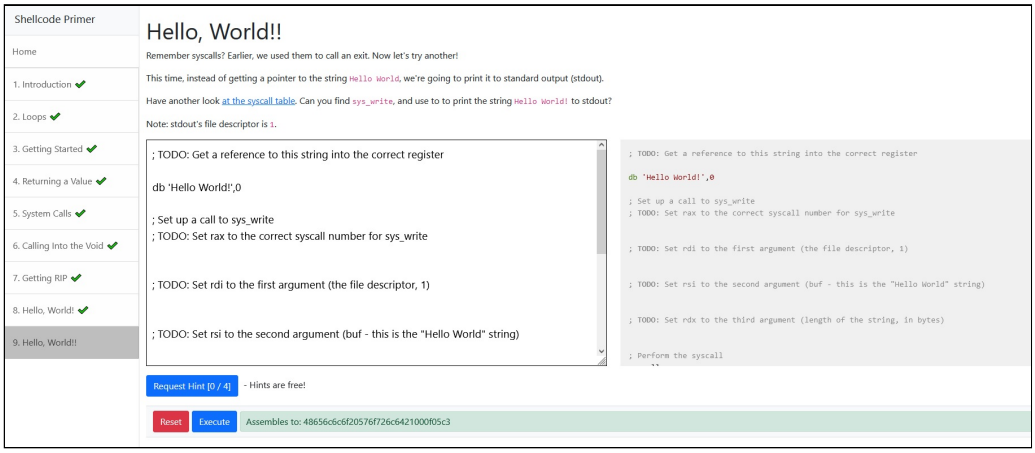
39. Modify the contents of the code window to

```
; This would be a good place for a call
call str_ref
; This is the literal string 'Hello World', null terminated, as code. Except
; it'll crash if it actually tries to run, so we'd better jump over it!
db 'Hello World',0

; This would be a good place for a label and a pop
str_ref:
pop rax
; This would be a good place for a re... oh wait, it's already here. Hooray!
ret
```

40. Click the `Execute` button

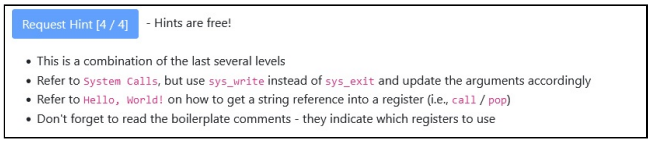
41. Close the Debugger window

42. Click `9. Hello, World!!`



43. Click the `Request Hint [0/4]` button and then click the `Request Hint [1/4]` button

44. Click the `Request Hint [2/4]` button and then click the `Request Hint [3/4]` button



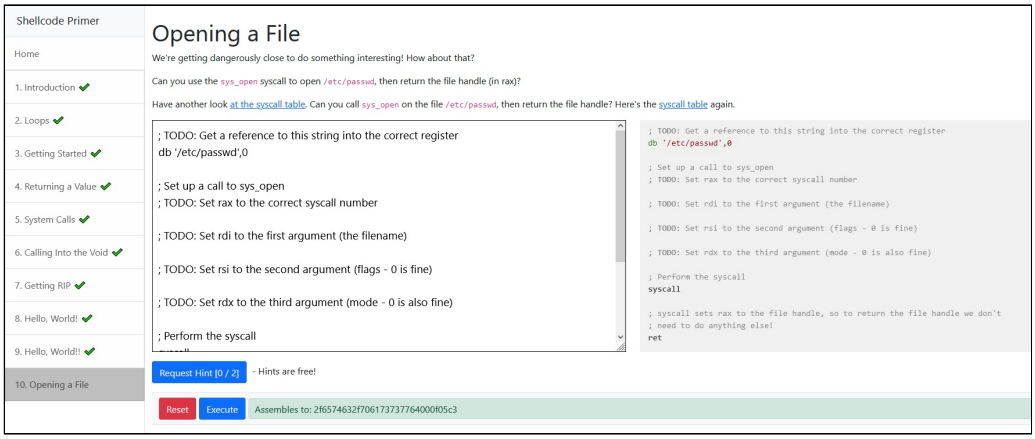
45. Modify the contents of the code window to

```
; TODO: Get a reference to this string into the correct register
call str_ref
db 'Hello World!',0
str_ref:
pop rbx
; Set up a call to sys_write
; TODO: Set rax to the correct syscall number for sys_write
mov rax, 1

; TODO: Set rdi to the first argument (the file descriptor, 1)
mov rdi, 1

; TODO: Set rsi to the second argument (buf - this is the "Hello World" string)
mov rsi, rbx

; TODO: Set rdx to the third argument (length of the string, in bytes)
mov rdx, 12

; Perform the syscall
syscall

; Return cleanly
ret
```

46. Click the `Execute` button

47. Close the Debugger window

48. Click `10. Opening a File`



49. Click the `Request Hint [0/2]` button and then click the `Request Hint [1/2]` button

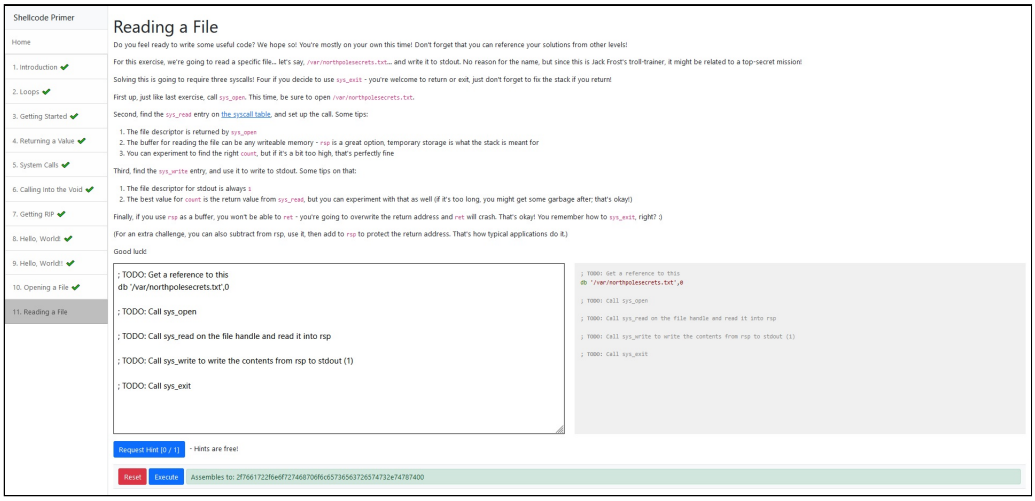
50. Modify the contents of the code window to

```
; TODO: Get a reference to this string into the correct register
call str_ref
db '/etc/passwd',0
str_ref:
pop rbx
; Set up a call to sys_open
; TODO: Set rax to the correct syscall number
mov rax, 2
; TODO: Set rdi to the first argument (the filename)
mov rdi, rbx
; TODO: Set rsi to the second argument (flags - 0 is fine)
mov rsi, 0
; TODO: Set rdx to the third argument (mode - 0 is also fine)
mov rdx, 0
; Perform the syscall
syscall

; syscall sets rax to the file handle, so to return the file handle we don't
; need to do anything else!
ret
```

51. Click the `Execute` button

52. Close the Debugger window

53. Click `11. Reading a File`

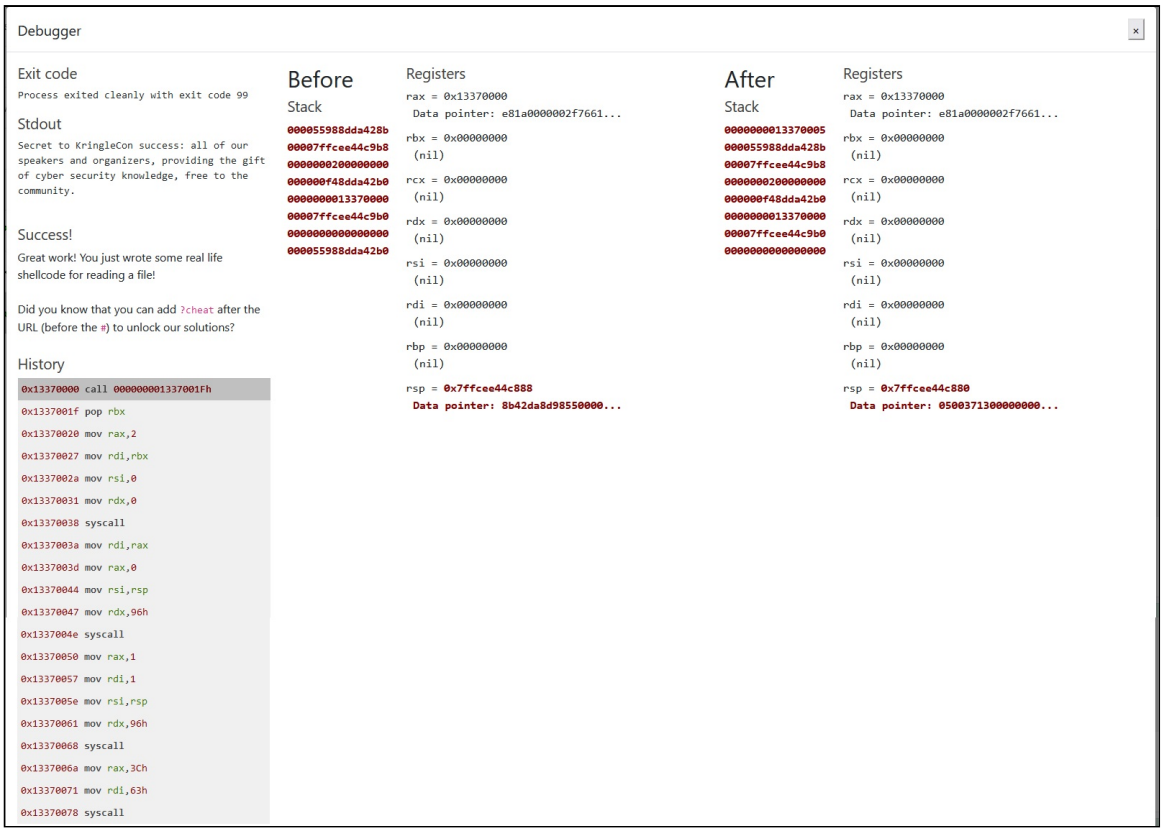


54. Click the `Request Hint [0/1]` button

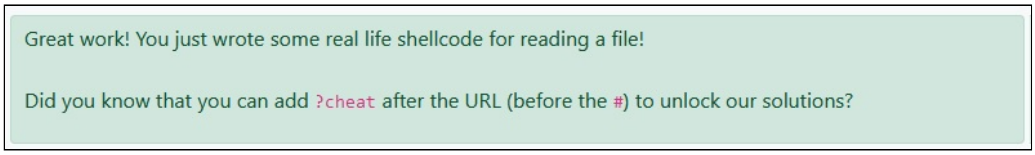
55. Modify the contents of the code window to

```
; TODO: Get a reference to this
call secret_ref
db '/var/northpolesecrets.txt',0
secret_ref:
pop rbx
; TODO: Call sys_open
mov rax, 2
mov rdi, rbx
mov rsi, 0
mov rdx, 0
syscall

; TODO: Call sys_read on the file handle and read it into rsp
mov rdi, rax
mov rax, 0
mov rsi, rsp
mov rdx, 150
syscall

; TODO: Call sys_write to write the contents from rsp to stdout (1)
mov rax,
mov rdi, 1
mov rsi, rsp
mov rdx, 150
syscall

; TODO: Call sys_exit
mov rax,  60
mov rdi, 99
syscall
```

56. Click the `Execute` button



57. Close the Debugger window

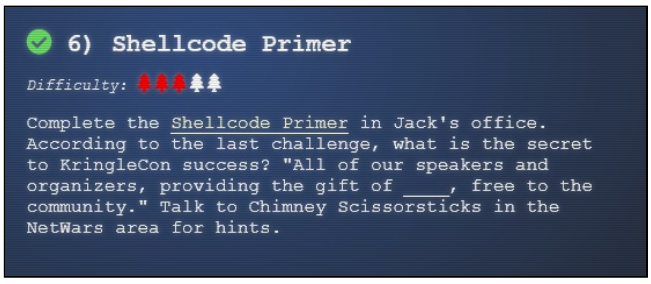


58. Close the `Shellcode Primer` and then click the `Tick` (Objectives) icon

59. Click `6) Shellcode Primer`

60. Type `cyber security knowledge`

61. Click the `Submit` button

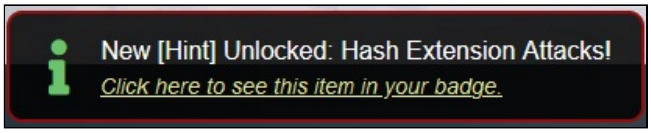


62. Click to talk to Ruby Cyster

> Oh man - what is this all about? Great work though.
>
> So first things first, you should definitely take a look at the firmware.
>
> With that in-hand, you can pick it apart and see what's there.

63. Click to talk to Ruby Cyster

> Did you know that if you append multiple files of that type, the last one is processed?
>
> Have you heard of Hash Extension Attacks?

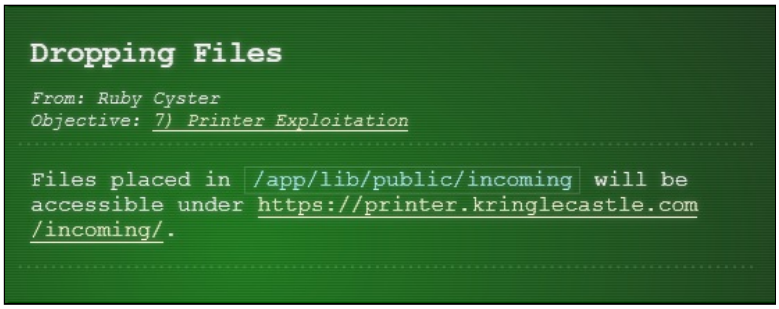


64. Click to talk to Ruby Cyster

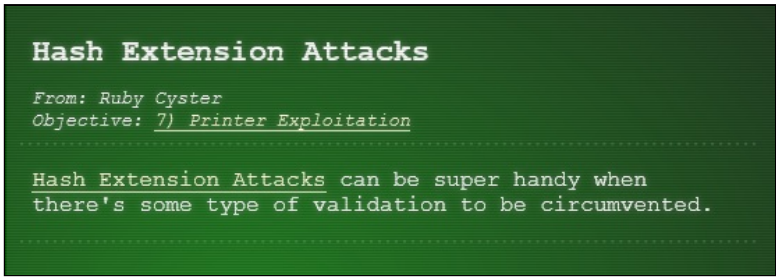
> If something isn't working, be sure to check the output! The error messages are very verbose.
>
> Everything else accomplished, you just might be able to get shell access to that dusty old thing!

65. Click the i (Hints) icon

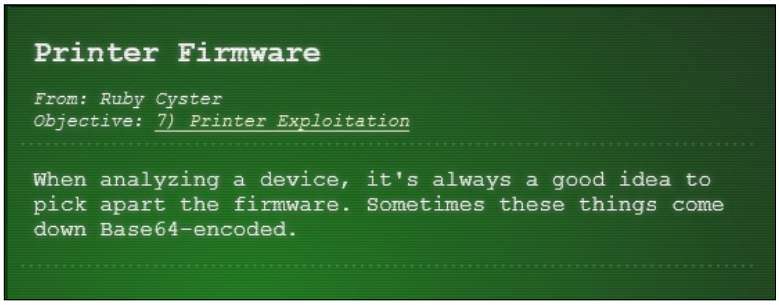
66. Click Dropping Files



67. Click `Hash Extension Attacks`

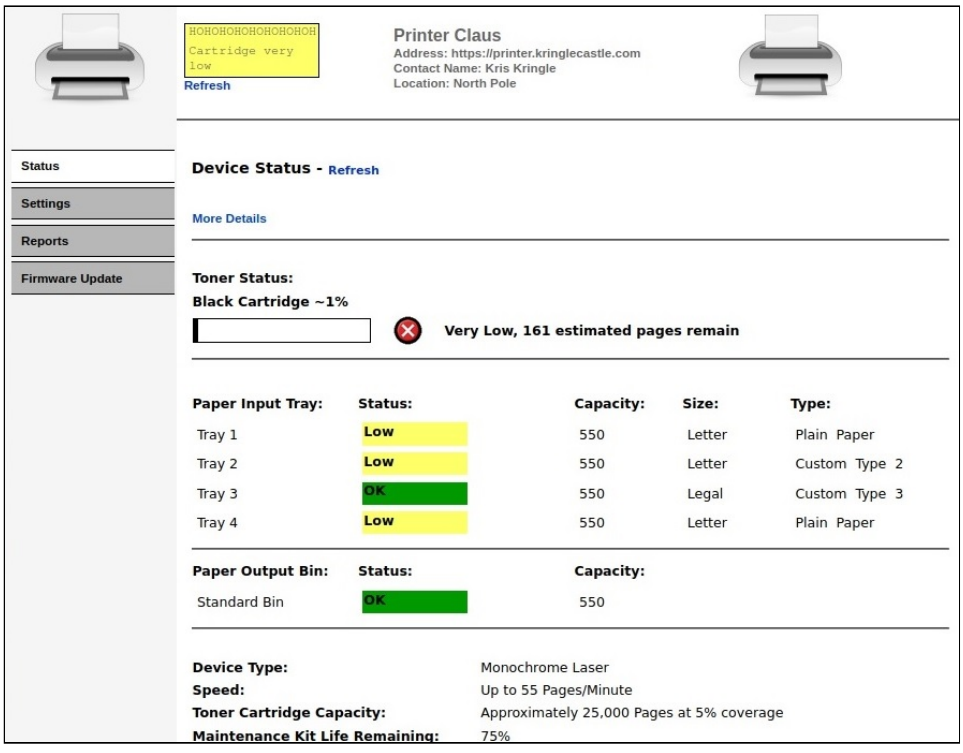


68. Click `Printer Firmware`



69. Click `[Exit]`

70. Click `Printer Exploitation`

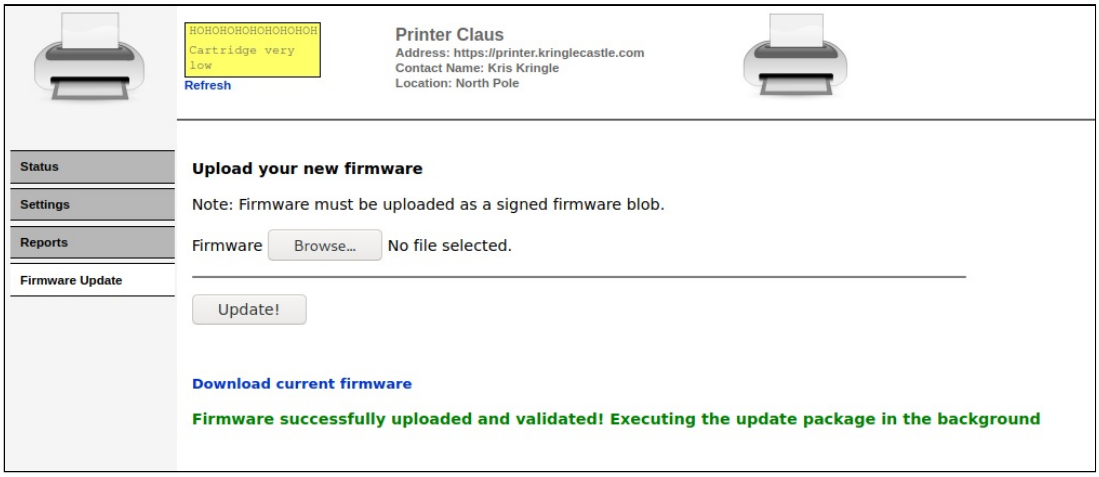


72. Click `Firmware Update`

73. Click the `Browse..` button

74. Select the `newfirmware.json` file

75. Click the `Update!` button

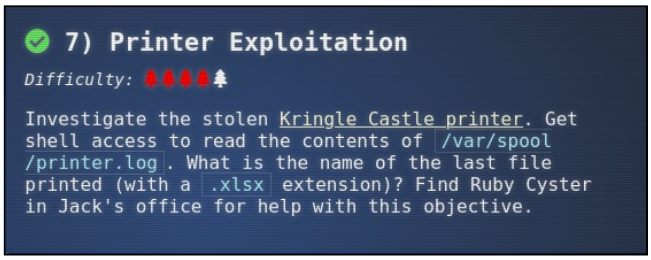


76. Click the `tick` (Objectives) icon

77. Click `7) Printer Exploitation`

78. Type `Troll_Pay_Chart.xlsx`

79. Click the `Submit` button



80. Click `[Exit]`

81. Move `Left` to enter the `Jack's Studio`