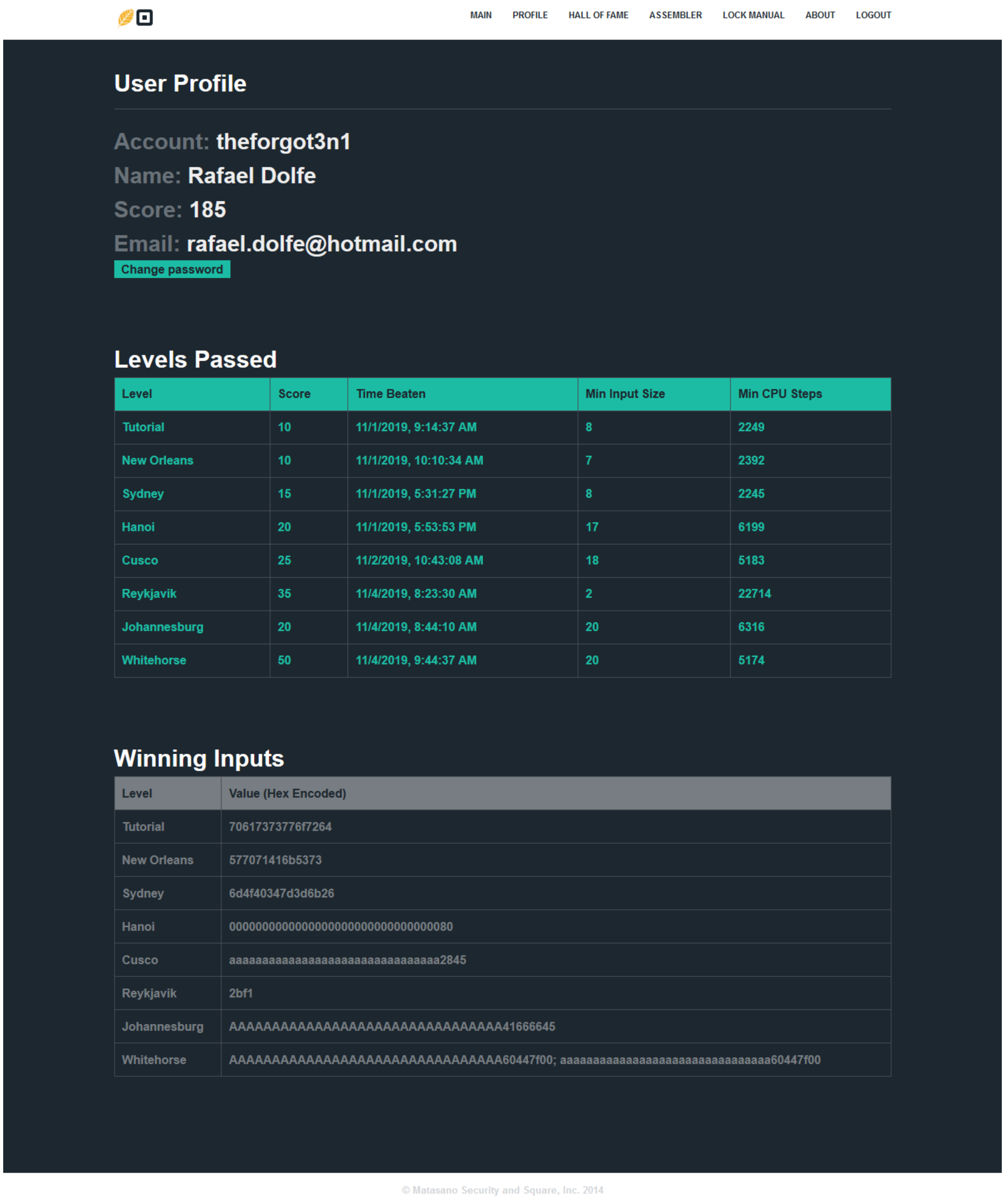
Microcorruption – results and why

1. New Orleans: The password is stored in memory after the password is created. Just write it down and then enter it as input. Answer: WpqAkSs
2. Sydney: Instead of storing the password in memory, Sydney instead “stores” the password as a couple of if-checks. Just satisfy the four cmp instructions and you have the password. Answer: mO@4}=k&
3. Hanoi: Thanks to the hint I was able to quickly find the instruction on line 4556 (cmp.b). What it does is compare 0x80 and whatever is on address 0x2410. So if I put 0x80 into address 0x2410 I have bypassed this compare instruction. So let’s do that by overflowing the input (putting 18 characters into the input). Answer in hex: 0000000000000000000000000000000080. All the 0s before 80 could be anything.
4. Cusco: This one took a long time. I did not see the return address of main being so vulnerable. I just tested random long inputs and it broke every time. When I realized the return address could be used to directly jump to unlocking the lock, I entered 4528 there. When it still didn’t work, I changed it to 2845 because of how it reads words. Tricky, many small details, but finally finished. Answer in hex: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa2845. All the a values could be anything.
5. Reykjavik: Basically the coders trying to confuse the attacker by adding a bunch of decryption/encryption stuff in the stack at 0x2400, and then running the stack there instead of having normal functions running. Took me ages to figure out what was actually happening at 0x2400 but eventually I figured out that the instruction cmp #0xf12b, -24(r4) was the thing that compared and checked for the right password. By making sure that cmp instruction was turning the zero flag to zero, I finished the level. Answer in hex: 2bf1.
6. Johannesburg: Same as Cusco except you have to bypass a check the lock has just before the return address. It is like a small canary, except the canary is checked against a static value, which can be used to trick the program. After you add 66 to make sure the comparison works out, just write your desired return address. Answer in hex: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA41666645
7. Whitehorse: This one went much quicker than many others, despite having a very complicated solution. At this point I had gotten a lot more of a grasp of how things worked. The solution combined several avenues of buffer overflow attacks – first the return address of main was overwritten. Second, the right flag was passed as an argument to the INT function (7f00) in order to unlock the lock regardless of password.
   Firstly, I implemented a return address jump to the address 4460, which is where the call to the interrupt was being made (call   #0x4532 <INT>). This was done the same way it was done previously, but without any canary in the way: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa6044. After this return address was manipulated to directly call <INT>, all that was needed was to set the correct flag to have the lock unlock the deadbolt without a password. For this I needed the microcorruption manual, where all possible arguments to the <INT> function are given. 0x7f is

the argument that will unlock the lock, without needing an argument. Since the <INT> function takes arguments through the stack, I merely added 0x7f00 after my return address manipulation. Thus my operatives were able to infiltrate the warehouse. Answer in hex: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa60447f00.

Screenshot of profile on microcorruption



User Profile

Account: theforgot3n1
Name: Rafael Dolfe
Score: 185
Email: rafael.dolfe@hotmail.com

Levels Passed

Level	Score	Time Beaten	Min Input Size	Min CPU Steps
Tutorial	10	11/1/2019, 9:14:37 AM	8	2249
New Orleans	10	11/1/2019, 10:10:34 AM	7	2392
Sydney	15	11/1/2019, 5:31:27 PM	8	2245
Hanoi	20	11/1/2019, 5:53:53 PM	17	6199
Cusco	25	11/2/2019, 10:43:08 AM	18	5183
Reykjavik	35	11/4/2019, 8:23:30 AM	2	22714
Johannesburg	20	11/4/2019, 8:44:10 AM	20	6316
Whitehorse	50	11/4/2019, 9:44:37 AM	20	5174

Winning Inputs

Level	Value (Hex Encoded)
Tutorial	70617373776f7264
New Orleans	577071416b5373
Sydney	6d4f40347d3d6b26
Hanoi	00000000000000000000000000000080
Cusco	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa2845
Reykjavik	2bf1
Johannesburg	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA41666645
Whitehorse	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA60447f00; aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa60447f00

© Matasano Security and Square, Inc. 2014