Application for Enhancing Software Reverse Engineering Learning

Jake Imanaka

Advisor: Mehdi Tarrit Mirakhorli

MS Plan B, Spring 2024

Motivation

Software Reverse engineering and binary exploitation are two cybersecurity skills necessary for understanding, discovering, and protecting against malware and software vulnerabilities. However, learning these skills can pose a significant challenge. In addition to any necessary prior knowledge in programming and operating systems, the tools commonly used in these endeavors can be complex and new-user unfriendly. Furthermore newcomers may need to install and learn multiple complex tools, program libraries and runtimes, and even multiple operating systems. This poses a substantial barrier to entry before effective reverse engineering and binary exploitation learning can take place.

Literature Review Snippet

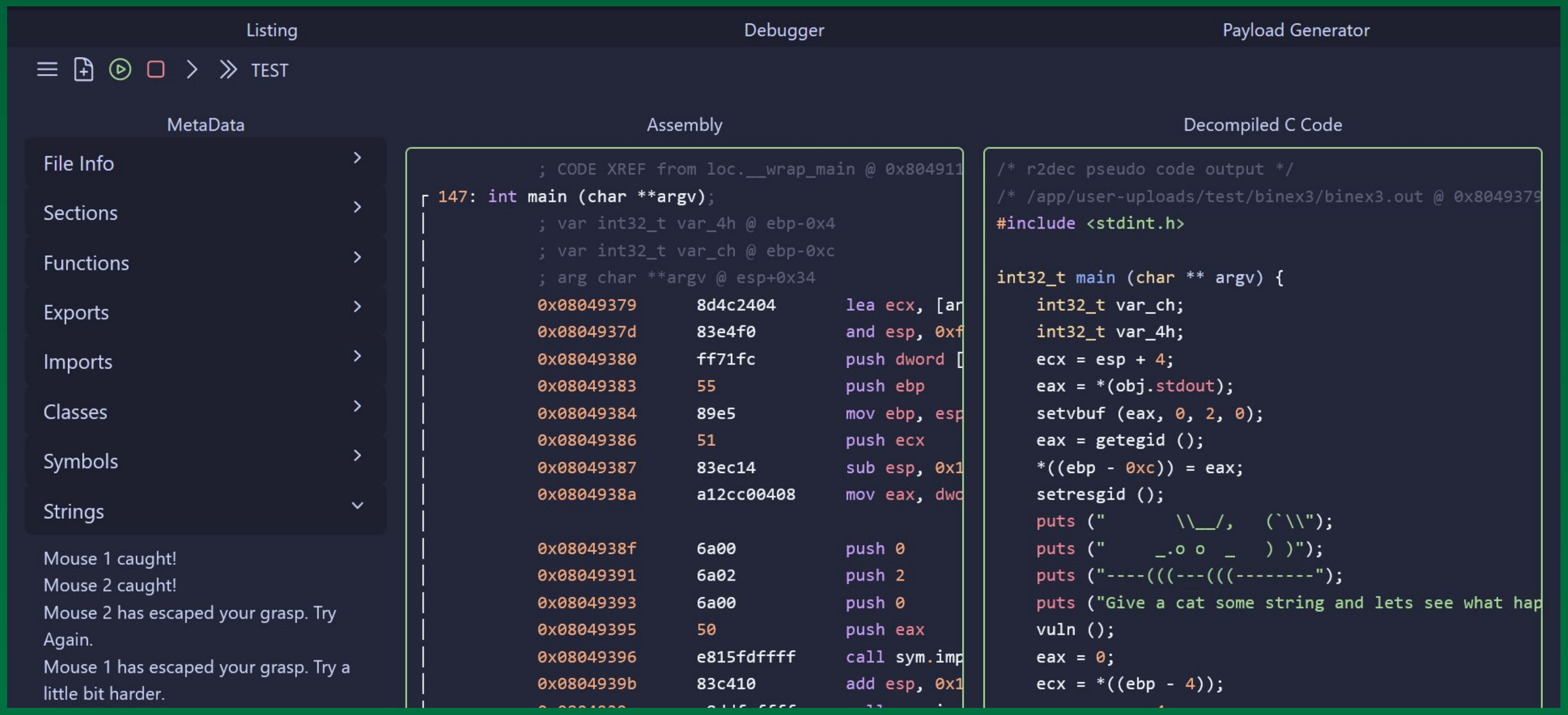
- Aycock et. al. states that “1 in 30 students are truly adept at reverse engineering” [2].
- Golden exclaims that there were “virtually no reverse engineering courses” partially due to the difficulty of developing curriculum [1].
- Mahoni and Ghandi reason that teaching/learning reverse engineering may be similar to learning art--”they either get it or they don’t” [3]

Proposed Solution and Goals

Web-based reverse engineering and binary exploitation tool with following goals:

- Cross platform compatibility
- No local installation required
- Seamless deployment
- Scalability and modularity
- Robust, but simplified environment
- Modern and unified UI
- Built-in learning environment

Disassembler & Decompiler

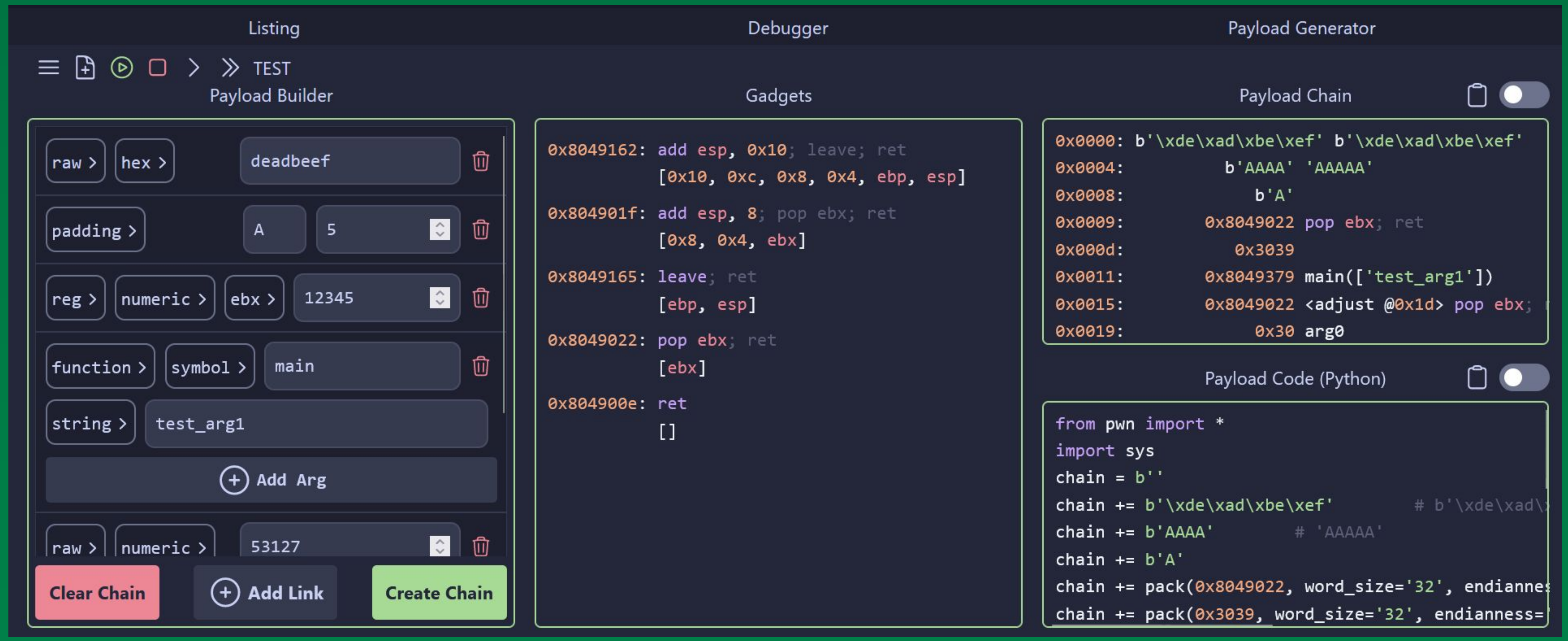


Mouse 1 caught!
Mouse 2 caught!
Mouse 2 has escaped your grasp. Try Again.
Mouse 1 has escaped your grasp. Try a little bit harder.

```
        ; CODE XREF from loc.__wrap_main @ 0x804911
┌ 147: int main (char **argv);
│       ; var int32_t var_4h @ ebp-0x4
│       ; var int32_t var_ch @ ebp-0xc
│       ; arg char **argv @ esp+0x34
│       0x08049379      8d4c2404        lea ecx, [ar
│       0x0804937d      83e4f0          and esp, 0xf
│       0x08049380      ff71fc          push dword [
│       0x08049383      55              push ebp
│       0x08049384      89e5            mov ebp, esp
│       0x08049386      51              push ecx
│       0x08049387      83ec14          sub esp, 0x1
│       0x0804938a      a12cc00408      mov eax, dwo
│
│       0x0804938f      6a00            push 0
│       0x08049391      6a02            push 2
│       0x08049393      6a00            push 0
│       0x08049395      50              push eax
│       0x08049396      e815fdffff      call sym.imp
│       0x0804939b      83c410          add esp, 0x1
```

```
/* r2dec pseudo code output */
/* /app/user-uploads/test/binex3/binex3.out @ 0x8049379
#include <stdint.h>

int32_t main (char ** argv) {
    int32_t var_ch;
    int32_t var_4h;
    ecx = esp + 4;
    eax = *(obj.stdout);
    setvbuf (eax, 0, 2, 0);
    eax = getegid ();
    *((ebp - 0xc)) = eax;
    setresgid ();
    puts ("       \\__/,   (`\\");
    puts ("     _.o o  _   ) )");
    puts ("----(((---(((--------");
    puts ("Give a cat some string and lets see what hap
    vuln ();
    eax = 0;
    ecx = *((ebp - 4));
```

Payload Generator



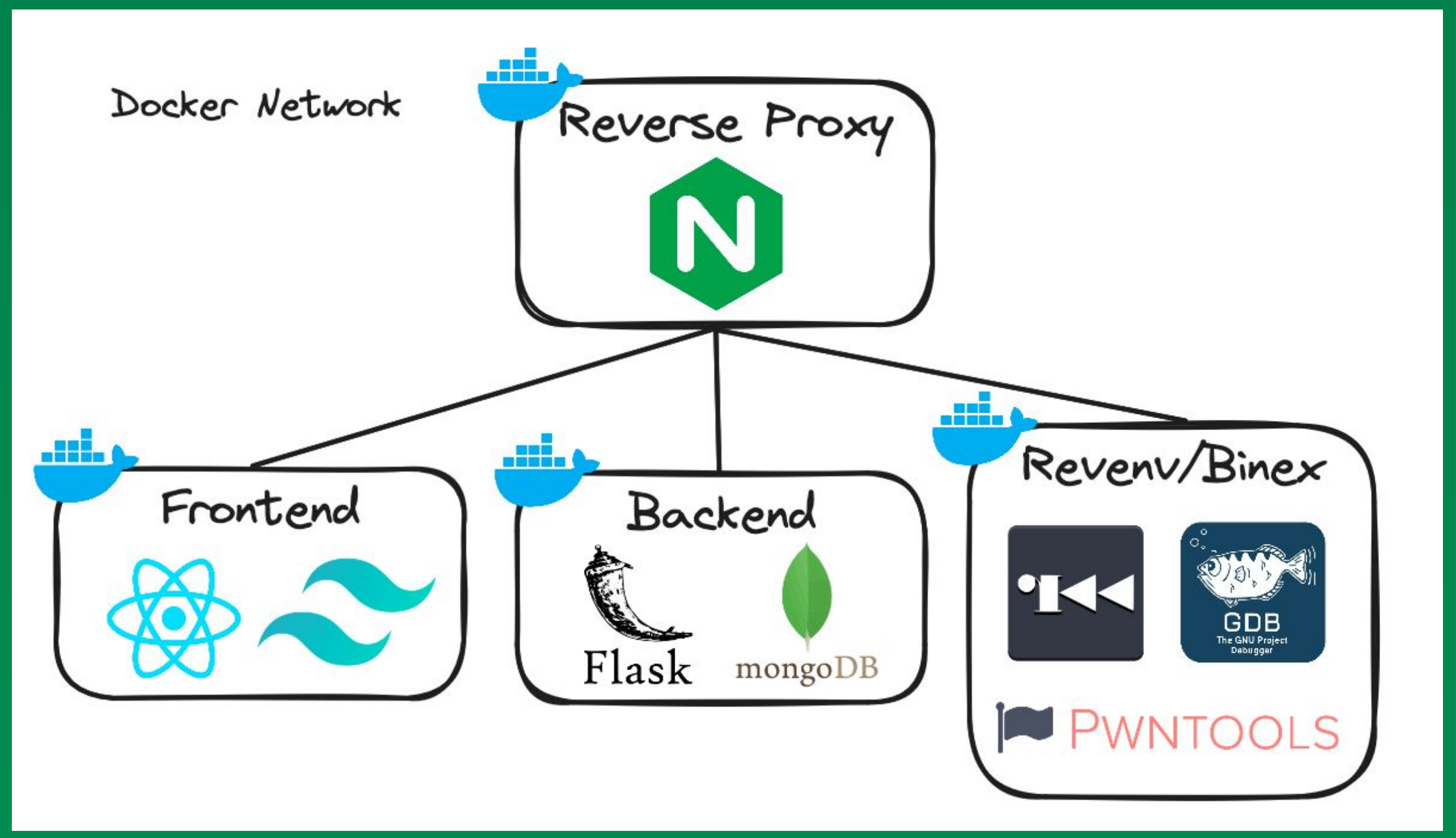
raw > hex > deadbeef
padding > A 5
reg > numeric > ebx > 12345
function > symbol > main
string > test_arg1
raw > numeric > 53127

```
0x8049162: add esp, 0x10; leave; ret
           [0x10, 0xc, 0x8, 0x4, ebp, esp]

0x804901f: add esp, 8; pop ebx; ret
           [0x8, 0x4, ebx]

0x8049165: leave; ret
           [ebp, esp]

0x8049022: pop ebx; ret
           [ebx]

0x804900e: ret
           []
```

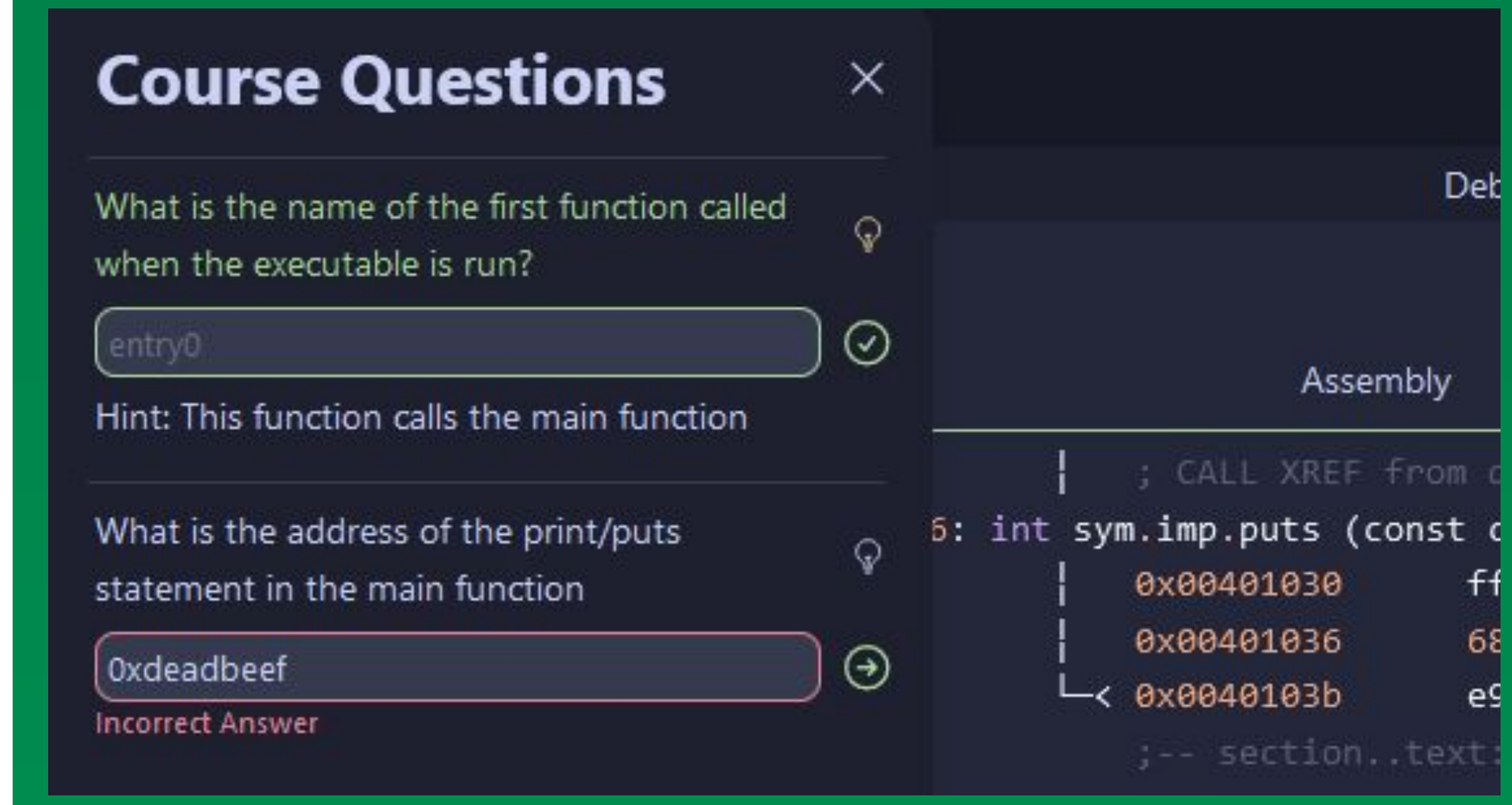
```
0x0000: b'\xde\xad\xbe\xef' b'\xde\xad\xbe\xef'
0x0004:           b'AAAA' 'AAAAA'
0x0008:              b'A'
0x0009:         0x8049022 pop ebx; ret
0x000d:            0x3039
0x0011:         0x8049379 main(['test_arg1'])
0x0015:         0x8049022 <adjust @0x1d> pop ebx;
0x0019:              0x30 arg0
```

```
from pwn import *
import sys
chain = b''
chain += b'\xde\xad\xbe\xef'        # b'\xde\xad\
chain += b'AAAA'          # 'AAAAA'
chain += b'A'
chain += pack(0x8049022, word_size='32', endiannes
chain += pack(0x3039, word_size='32', endianness=
```

Tech Stack



Lesson Integration



What is the name of the first function called when the executable is run?

entry0

Hint: This function calls the main function

What is the address of the print/puts statement in the main function

0xdeadbeef

Incorrect Answer

```
    ; CALL XREF from c
5: int sym.imp.puts (const c
    0x00401030      ff
    0x00401036      68
└─< 0x0040103b      e9
    ;-- section..text:
```

Results

Reverse Nexus--A web-based software reverse engineering and binary exploitation platform for novices and educators.

- File header and binary analysis
- Binary disassembly
- Binary decompilation
- Live debugging
- Binary exploitation payload builder with debugging integration
- integrated guided lesson creation and registration
- One-click deployment and modularity with Docker stacks

Conclusion

Reverse Nexus improves the process of learning reverse engineering and binary exploitation by lowering their barrier to entry. This is accomplished by providing a simplified full-featured environment for both students and educators. Students can benefit by not needing to learn multiple complex user interfaces, locally install various tools, find practice binaries and lessons by themselves, and install various virtual machines. Educators can benefit by utilizing the tool’s automated setup and deployment via Docker.

Future Work

Future development goals include: course filtering, additional course metadata, supporting more CPU architectures, and implementing more UI functionalities for the integrated debugger. Additionally, as this project encapsulates only the first phase of development, user and group testing should be performed to evaluate the usefulness and feasibility of the tool. This may be a topic for a future research paper or thesis

References

[1] R. G. Golden III, “A Highly Immersive Approach to Teaching Reverse Engineering.” Accessed: Apr. 06, 2024. [Online]. Available: https://www.usenix.org/legacy/event/cset09/tech/full_papers/richard.pdf

[2] J. Aycock, A. Groeneveldt, H. Kroepfl, and T. Copplestone, “Exercises for teaching reverse engineering,” Jul. 2018, doi: https://doi.org/10.1145/3197091.3197111.

[3] W. Mahoney and R. A. Gandhi, “Reverse engineering - Is It Art?” ACM Inroads, vol. 3, no. 1, pp. 56–61, Mar. 2012, doi: https://doi.org/10.1145/2077808.2077826.