BREAKING DECOMPILERS

...or, How To Make Jordan's Life Hard

BINARYNINJA

YOU?

Who has...

- written C?
- used a debugger?
- used a decompiler?
- written a decompiler?
- written a decompiler plugin?

GOALS

After this talk, you should:

- understand more about how decompilers work and thus,
- have lots of ideas on how to break them

ANTI-GOALS

- Not breaking debuggers
- Not an exhaustive list of all possible techniques

OUTLINE

- Why Decompilers Are Impossible
- How Decompilers Work At All
- (ノ°□°　ノ　　┻━┻

WHY DECOMPILERS ARE IMPOSSIBLE

INFORMALLY

- Comments
- Symbol Names (in a stripped binary without debug)

INFORMALLY

```c
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
  int fd = open(argv[1], O_RDONLY);
  size_t size = lseek(fd, 0, SEEK_END);
  int prot = PROT_EXEC | PROT_READ;
  void *mem = mmap(NULL, size, prot, MAP_PRIVATE, fd, 0);
  ((void(*)())mem)();
  return 0;
}
```

FORMALLY

https://en.wikipedia.org/wiki/Rice's_theorem

BUT...

HOW DECOMPILERS WORK AT ALL

HOW DECOMPILERS WORK AT ALL

- Parsing
- Lifting
- Optimizing

HOW ~~DE~~COMPILERS WORK

- Parsing
- ~~Lifting~~ Lowering
- Optimizing

Summarize how a binary compiler works in three steps.

1. **Parsing and Lexical Analysis**: The source code is parsed to check for syntax errors and is broken down into tokens by the lexical analyzer. These tokens are the basic building blocks, like keywords, operators, and identifiers.

2. **Intermediate Code Generation**: The tokens are converted into an intermediate representation (IR), which is a lower-level code that is easier for the compiler to optimize and translate into machine code.

3. **Code Generation and Optimization**: The intermediate code is optimized for performance and then translated into machine code specific to the target architecture. This machine code is then output as a binary executable file.

PARSING

Universal:

- Memory Mappings
- Entry Point
- Exports / Imports
- Control Flow / Code Discovery

PARSING

Specific:

- Mach-O / PE / ELF
- Section Information
- Metadata
- Relocations

LIFTING

- Translating from native to intermediate
- BNIL, Microcode, P-Code
- See BlueHat Talk or Updated Slides

OPTIMIZING

- Applying type information
- Matching signatures
- Dead Code Elimination
- Resolve Indirect Control Flow
- Higher Level Control Flow Structures

EVALUATION CRITERIA

EFFECTIVE

How much does it prevent analysis/understanding?

EVIDENT

How obvious is it?

EFFORT

How much work is it to implement?

(ノ °□°) ノ ⌒ ┻━┻

BASE CHALLENGE

```c
#include <stdio.h>
#include <string.h>

int main() {
    char input[100];
    printf("Enter the password: ");
    fgets(input, sizeof(input), stdin);
    if (strcmp(input, "correct") == 0) {
        printf("Access granted!\n");
    } else {
        printf("Access denied!\n");
    }
    return 0;
}
```

EXAMPLES

- Break Parsing
- Segments
- Relocations
- Break Lifting
- Alignment
- Vectorization
- Break Optimizations
- What's in a Name?
- Packers
- Custom Compiler
- Permissions/Dataflow

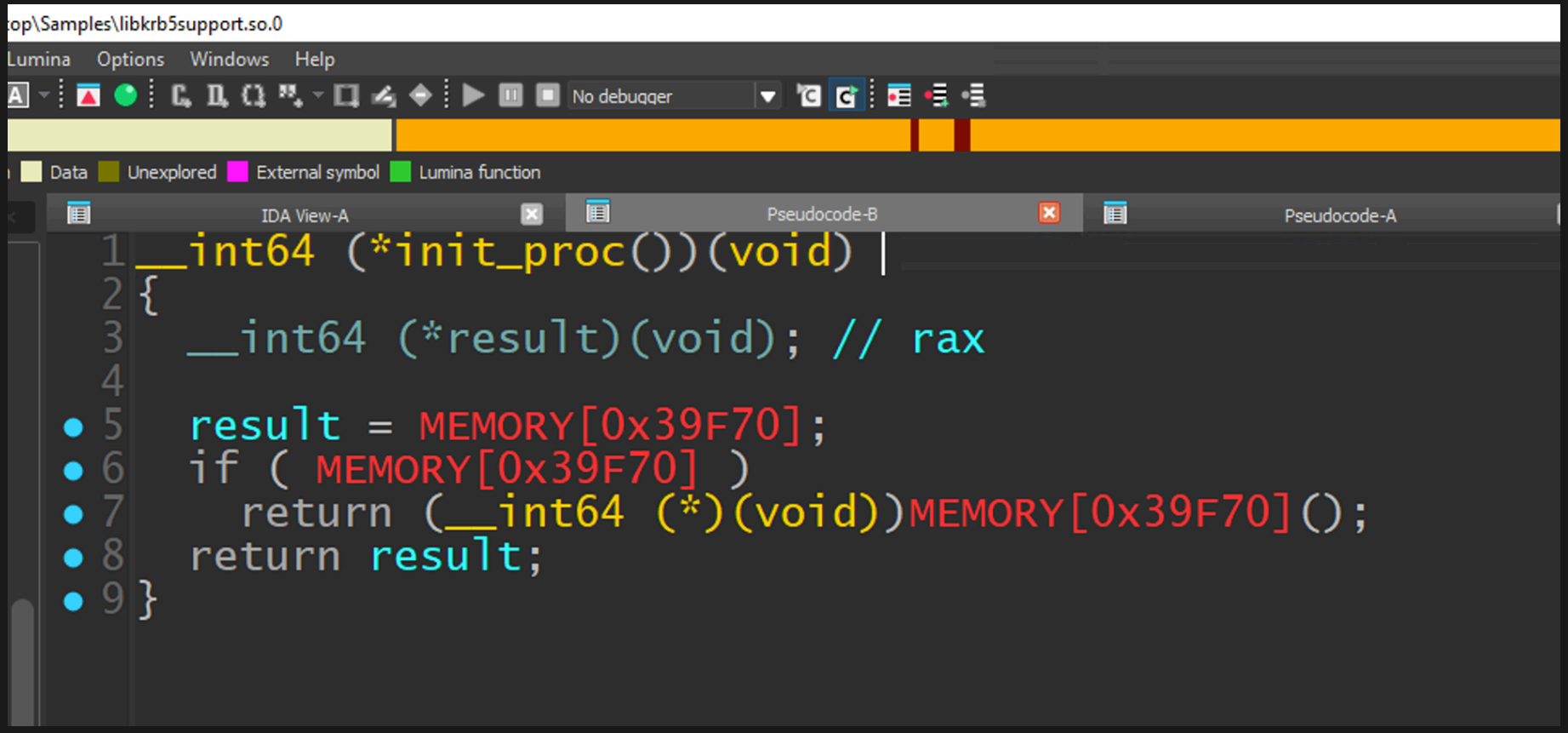
BREAK THE PARSING

SEGMENT SHENANIGANS

	Effective	5
	Evident	5
	Effort	1

DEMO!

`./examples/zetatwo`


op\Samples\libkrb5support.so.0
Lumina Options Windows Help
No debugger
Data Unexplored External symbol Lumina function
IDA View-A
Pseudocode-B
Pseudocode-A
1 __int64 (*init_proc())(void)
2 {
3 __int64 (*result)(void); // rax
4
5 result = MEMORY[0x39F70];
6 if (MEMORY[0x39F70])
7 return (__int64 (*)(void))MEMORY[0x39F70]();
8 return result;
9 }

RELOCATIONS

Relocations are the worst

	Effective	5
	Evident	5
	Effort	1

BUILD YOUR OWN!

1. Fuzz the file, run it.
2. If it still works, dump the decompilation and pattern match
3. GOTO 1

BREAK THE LIFTING

ALIGNMENT

	Effective	3
	Evident	2
	Effort	5

DEMO!

`./examples/alignment`

VECTORIZED

Just use an instruction that is rare and not implemented, or is incorrectly lifted.

 Effective	3
 Evident	2
 Effort	2

BREAK THE OPTIMIZATIONS

STOP 🛑

	Effective	3
	Evident	2
	Effort	2

DEMO!

```
./examples/stop
```

UPX

 Effective	4
 Evident	4
 Effort	5

```
./examples/upx
```

DEMO!

SCC

Our newly open-sourced Shellcode Compiler supports many built in obfuscations.

Effective 4

Evident 2

Effort 5

SCC

DEMO!

DATAFLOW PROPAGATION/MEMORY PERMISSIONS

	Effective	4
	Evident	4
	Effort	5

DEMO!

`./examples/perms`

SCC

	Effective	4
	Evident	4
	Effort	5

DEMO!

`./examples/scc`

SUMMARY OF TECHNIQUES

Technique	Effectiveness	Evident	Effort
Segment Shenanigans	5	5	1
Relocations	5	5	1
Alignment	3	2	5
Vectorized	3	2	2
STOP	3	2	2
UPX	4	4	5
SCC	4	2	5
Dataflow/Permissions	4	4	5

CONCLUSION

- What are your goals?
- Do you care more about increased difficulty or subtle breakage?
- Decompilers are easy to break, hard to make.

QUESTIONS?

- https://github.com/psifertex/breaking_decompilers
- https://twitter.com/psifertex
- https://binary.ninja/

CREDITS / ACKNOWLEDGEMENTS

- reveal-md
- podman
- chatgpt (o4-mini for image generation)