

Binlex

A Genetic Binary Traits Lexer



2021-11-22



terminal

```
c3rb3ru5@d3d53c $ whois c3rb3ru5d3d53c
```

Job: Director of Malware Lab

Experience: 6+ years

Skills: Reversing Engineering, Malware Analysis, C++, C, Assembly, Python

Hobbies: Guitar, Machine Learning

Twitter: <https://twitter.com/c3rb3ru5d3d53c>

GitHub: <https://github.com/c3rb3ru5d3d53c>



- ▶ Hunting Code-Reuse
- ▶ Writing YARA Signatures based on Code
- ▶ Binary Genetic Trait Lexer
- ▶ Machine Learning Feature Collection

Binlex

What is a Lexer?



lexer.log

In computer science, lexical analysis, lexing or tokenization is the process of converting a sequence of characters into a sequence of tokens.



terminal

```
c3rb3ru5@d3d53c $ git clone https://github.com/c3rb3ru5d3d53c/binlex.git
```

```
c3rb3ru5@d3d53c $ cd binlex/
```

```
c3rb3ru5@d3d53c $ sudo apt install -y git libcapstone-dev cmake make
```

```
c3rb3ru5@d3d53c $ sudo apt install -y parallel
```

```
c3rb3ru5@d3d53c $ make threads=4
```

```
c3rb3ru5@d3d53c $ sudo make install
```



terminal

```
c3rb3ru5@d3d53c $ binlex -help
binlex v1.0.0 - A Binary Genetic Traits Lexer
-i -input input file (required)
-m -mode set mode (required)
-lm -list-modes list modes
-h -help display help
-o -output output file
-v -version display version
Author: @c3rb3ru5d3d53c
```



terminal

```
c3rb3ru5@d3d53c $ binlex -lm
```

```
elf:x86
```

```
elf:x86_64
```

```
pe:x86
```

```
pe:x86_64
```

```
raw:x86
```

```
raw:x86_64
```

```
raw:cil
```

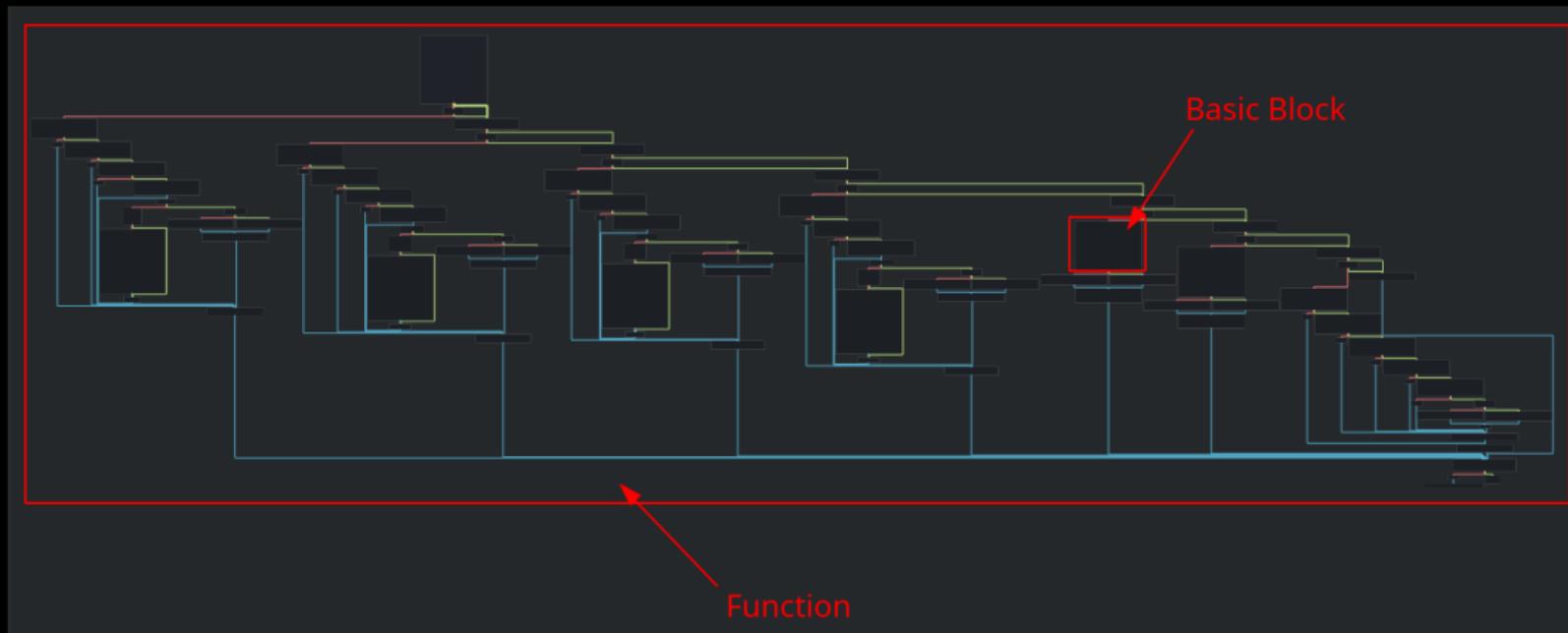


genetic_traits.log

Genetic binary traits consist of byte sequences. These byte sequences consist of hexadecimal strings with memory operands and others subject to change wild carded with question marks.

Binlex

What Makes a Trait?



Binlex

What Makes a Trait?



- ▶ Basic Blocks
 - ▶ jmp
 - ▶ jne
 - ▶ jz
- ▶ Functions
 - ▶ ret



terminal

```
c3rb3ru5@d3d53c $ binlex -m pe:x86 -i tests/pe/pe.x86 | head -2  
50 b9 60 44 41 ?? c7 00 40 05 41 ?? e8 78 46 ?? ?? c3  
c7 41 ?? e0 e0 e0 ?? 89 51 ?? 39 50 ?? 56 8b 74 24 ?? 74 12
```



- ▶ Current Tools
 - ▶ VirusTotal Diff
 - ▶ Intezer
 - ▶ Yargen
 - ▶ yara-signator
- ▶ Differences
 - ▶ Wildcarding
 - ▶ Framework
 - ▶ Write your own Logic
 - ▶ Not a Blackbox, It's OPEN!

Binlex

Build your Own!



imgflip.com



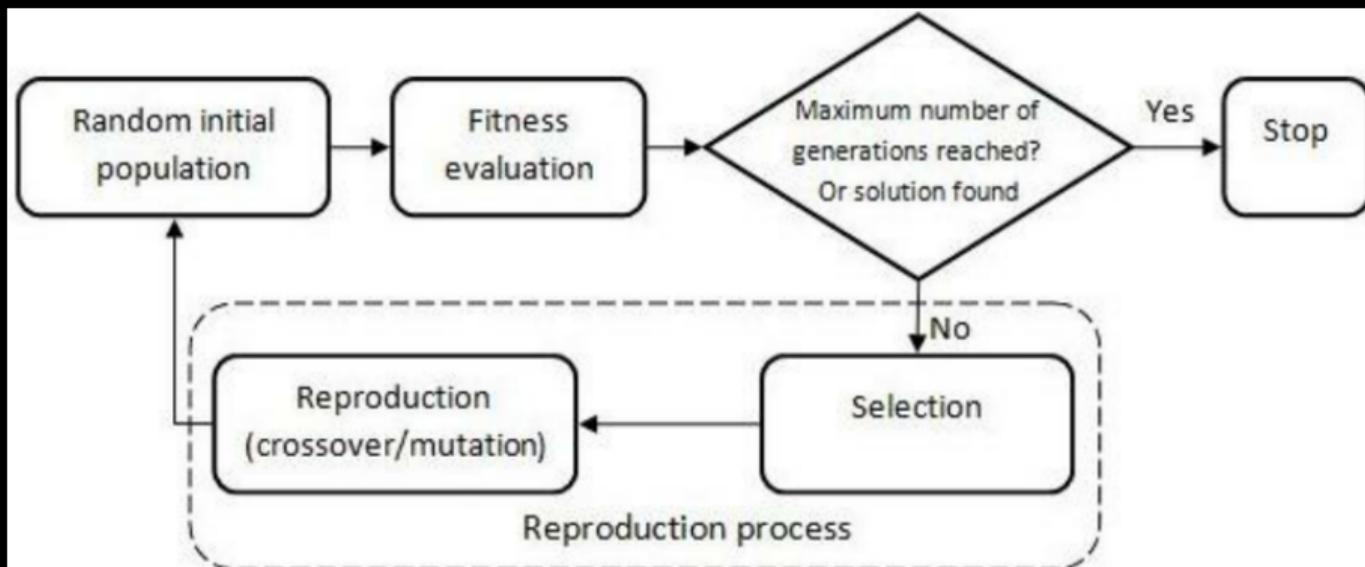
genetic_programming.log

In artificial intelligence, genetic programming (GP) is a technique of evolving programs, starting from a population of unfit (usually random) programs, fit for a particular task by applying operations analogous to natural genetic processes to the population of programs.



fitness.log

A fitness function is a particular type of objective function that is used to summarise, as a single figure of merit, how close a given design solution is to achieving the set aims. Fitness functions are used in genetic programming and genetic algorithms to guide simulations towards optimal design solutions.



Binlex

Building your Detection Logic - With Bash



```
c3rb3ru5 at d3d53c in ~/Tools/binlex/samples (master v)
λ find karma/ -type f -not -name "*.traits" | while read i; do binlex -m pe:x86 -i $i | sort | uniq; end | sort | uniq -c | sort -rn | head -10
7 8b f1 eb 02
7 83 c7 02 eb c5
7 66 83 fa 2e 75 06
7 66 83 fa 20 74 0a
7 5f 5e 33 c0 5b 8b e5 5d c3
7 33 f6 0f b7 41 ?? 83 c1 02 8b d0 66 85 c0 75 da
7 0f b7 d0 66 83 fa 5c 74 10
6 0f b7 14 06 2b d1 75 09
5 8b c1 59 94 8b ?? 89 04 24 f2 c3 2d 00 10 ?? ?? 85 ?? eb e7
5 83 ef 01 74 05
c3rb3ru5 at d3d53c in ~/Tools/binlex/samples (master v)
λ
```

Get traits from each sample

Find common traits for each sample

Occurrences of trait

0 (0.515s) < 16:42:28

Binlex

Building your Detection Logic - CIL/.NET



```
binlex: fish — Konsole
File Edit View Bookmarks Settings Help
c3rb3ru5 at d3d53c ln ~/Tools/binlex (master v)
λ hexdump -C samples/cil_bin
0 (0.003s) < 12:00:29
00000000 00 16 28 1c 00 00 0a 0a 72 f5 01 00 70 28 35 00 |..(.....r...p(5.|
00000010 00 0a 0b 07 72 0d 02 00 70 28 36 00 0a 0c 06 |...r...p(6.....|
00000020 72 59 01 00 70 28 1d 00 00 0a 72 e9 02 00 70 17 |rY..p(.....r...p.|
00000030 28 3e 00 00 0a 0d 08 72 59 01 00 70 28 1d 00 00 |(>.....rY..p(....|
00000040 0a 72 e9 02 00 70 17 28 3e 00 00 0a 13 04 73 22 |.r...p(>.....s*~|
00000050 00 00 06 13 05 72 ed 02 00 70 13 06 16 13 07 2b |.....r...p.....+|
00000060 16 00 11 05 09 11 07 9a 11 06 6f 21 00 00 06 00 |.....o!.....|
00000070 00 11 07 17 58 13 07 11 07 09 Be 69 fe 04 13 08 |...X.....t.....|
00000080 11 08 2d dd 16 13 09 2b 17 00 11 05 11 04 11 09 |...-...+.....|
00000090 9a 11 06 6f 21 00 00 06 00 00 11 09 17 58 13 09 |...o!.....X...|
000000a0 11 09 11 04 8e 69 fe 04 13 0a 11 0a 2d 0b 2a |...t.....-.*|
000000af
c3rb3ru5 at d3d53c ln ~/Tools/binlex (master v)
λ binlex -m rawcil -i samples/cil_bin
0 (0.001s) < 12:00:30
00 16 28 ?? ?? ?? ?? 0a 72 ?? ?? ?? ?? 28 ?? ?? ?? ?? 0b 07 72 ?? ?? ?? ?? 28 ?? ?? ?? ?? 0c 06 72 ?? ?? ?? ??
28 ?? ?? ?? ?? 72 ?? ?? ?? ?? 17 28 ?? ?? ?? ?? 0d 08 72 ?? ?? ?? ?? 28 ?? ?? ?? ?? 72 ?? ?? ?? ?? 17 28 ?? ??
?? ?? 13 ?? 73 ?? ?? ?? ?? 13 ?? 72 ?? ?? ?? 13 ?? 72 ?? ?? ?? 16 13 ?? 72 ?? 28 ?? ?? ?? 09 11 ?? 9a 11 ?? 6f ?? ?? ??
00 00 11 ?? 17 58 13 ?? 11 ?? 09 8e 69 fe 04 13 ?? 11 ?? 2d ?? 16 13 ?? 2b ?? 00 11 ?? 11 ?? 11 ?? 9a 11 ?? 6f
?? ?? ?? ?? 00 00 11 ?? 17 58 13 ?? 11 ?? 11 ?? 8e 69 fe 04 13 ?? 11 ?? 2d ?? 2a
00 16 28 ?? ?? ?? ?? 0a 72 ?? ?? ?? ?? 28 ?? ?? ?? ?? 0b 07 72 ?? ?? ?? ?? 28 ?? ?? ?? ?? 0c 06 72 ?? ?? ?? ??
28 ?? ?? ?? ?? 72 ?? ?? ?? ?? 17 28 ?? ?? ?? ?? 0d 08 72 ?? ?? ?? ?? 28 ?? ?? ?? ?? 72 ?? ?? ?? ?? 17 28 ?? ??
?? ?? 13 ?? 73 ?? ?? ?? ?? 13 ?? 72 ?? ?? ?? 13 ?? 72 ?? ?? ?? 16 13 ?? 72 ??
00 11 ?? 09 11 ?? 9a 11 ?? 6f ?? ?? ?? ?? 00 00 11 ?? 17 58 13 ?? 11 ?? 09 8e 69 fe 04 13 ?? 11 ?? 2d ??
16 13 ?? 2b ??
00 11 ?? 11 ?? 11 ?? 9a 11 ?? 6f ?? ?? ?? ?? 00 00 11 ?? 17 58 13 ?? 11 ?? 11 ?? 8e 69 fe 04 13 ?? 11 ?? 2d ??
2a
c3rb3ru5 at d3d53c ln ~/Tools/binlex (master v)
λ
```

.NET CIL Byte-Code

YARA Strings / Traits

.NET CIL Instructions



```
if (strcmp(args.options.mode, (char *)"raw:cil") == 0 &&
    args.options.io_type == ARGS_IO_TYPE_FILE){
    Raw raw_cil;
    if (raw_cil.ReadFile(args.options.input, 0) == false){
        return 1;
    }
    CILDecompiler cil_decompiler;
    if (cil_decompiler.Setup(CIL_DECOMPILER_TYPE_BLOCKS) == false){
        return 1;
    }
    if (cil_decompiler.Decompile(raw_cil.sections[0].data, raw_cil.sections[0].size, 0) == false){
        return 1;
    }
    if (cil_decompiler.Setup(CIL_DECOMPILER_TYPE_FUNCS) == false){
        return 1;
    }
    if (cil_decompiler.Decompile(raw_cil.sections[0].data, raw_cil.sections[0].size, 0) == false){
        return 1;
    }
    if (args.options.output == NULL){
        cil_decompiler.PrintTraits();
    } else {
        cil_decompiler.WriteTraits(args.options.output);
    }
    return 0;
}
```



- ▶ Cutter Plugin
- ▶ Ghidra Plugin
- ▶ IDA Plugin
- ▶ C++ API Documentation
- ▶ Python Bindings
- ▶ Machine Learning
- ▶ Default Fitness Function (built-in)
- ▶ Looking for Developers



- ▶ https://en.wikipedia.org/wiki/Fitness_function
- ▶ https://en.wikipedia.org/wiki/Genetic_programming
- ▶ <https://www.capstone-engine.org/>

Binlex

Demo Time!

