Level 0x0a

Binary Ninja

Topics

- Events
- Shirts

Code Quest

- Saturday, February 24
- 2.5 hours. Free breakfast and lunch
- Teams are 2-3 students
  - 1 laptop per person
  - Novice division
  - Advanced division (1 programmer with 1 year programming exp)
- High school students, ages 13-18 years old
- Parent Meeting / Transportation Discussion: Fri Feb 16, 3:45 PM, Library

Wildcat Coders 1	Wildcat Coders 2	Wildcat Coders 3	Wildcat Coders 4
Novice	Advanced	Novice	Advanced
- Eshan V - Saaketh K - Shoaib A	- Justin T - Dylan G - Sam S	- Gabriella Z - Dominic M - Avery M	- Jay J - Parker W - Zachary W

Upcoming Events

LOCKHEED MARTIN CYBERQUEST® COMPETITION

- Lockheed Martin Cyber Quest
  - Saturday, March 23rd
  - 3 hours. Free Breakfast and Lunch
  - Teams are 3-5 students
    - 3 laptops per team
  - No team limit given
  - Registration Timeframe:
    - Wed Jan 03 2024 - Mon Feb 26 2024
- Pico CTF (Carnagie Mellon)
  - March 12-26
  - Online CTF

GenCyber Camp

- Cyber Security Summer Camp at Florida Tech
- 60 hours of instruction / labs
- June 10 - 14 - Register at https://event.fit.edu/gencyber
- 40 students, entering 9th - 12th grades
- Big Brother CTF - March 2nd 10AM - 4 PM
- Student selection is on Monday Feb 19th, so register before then!!
- Funded via NSA grant, no cost to students

Cyber Fundamentals Course

- The fall course gave out scholarship to winning student
- After school class for 5 weeks in April
- Tuesdays and Thursdays
- Very early planning, stay tuned

Modern Vintage Gamer (Dimitris Giannakis)

- Xbox Homebrew Developer (Lantus)
  - Ported Doom, Quake, Quake 2 to XBox
- Youtube series highlights
  - Mistakes were made (DRM and gaming)
  - Game preservation
  - Console hacking
- Works now as developer / game ports for Switch / PS4 / PS5

Static Vs Dynamic Linking

- `ldd` - Prints the shared objects / libraries required by an ELF binary
  - dumpbin might be the equivalent for Windows PE files
- Many programs share libraries so the binaries aren't bloated with libraries that everybody uses over and over
  - Libc: the basic C library
  - Libstdc++: basic library for most C++ functions / classes
  - Qt, GTK, Visual C++ Redistributable: basic libraries for desktop GUI applications
  - OpenGL, DirectX, SDL: graphics libraries
- These are typically .so files that your system needs before it can run application
  - .dll for Windows systems

ldd examples

C application

```
mwales@Metroid:~/scratch/c_test$ ldd a.out
        linux-vdso.so.1 (0x00007ffe5d5f3000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x000
        /lib64/ld-linux-x86-64.so.2 (0x00007f9d202a3000)
```

C++ application

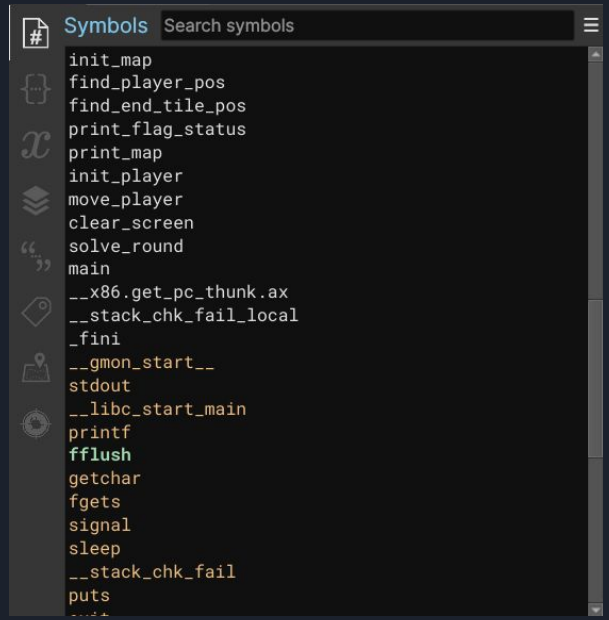
```
mwales@Metroid:~/scratch/cpp_test$ g++ hw.cpp
mwales@Metroid:~/scratch/cpp_test$ ldd ./a.out
        linux-vdso.so.1 (0x00007fff103dc000)
        libstdc++.so.6 => /lib/x86_64-linux-gnu/libstdc++.so.6 (0x006
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fb218e06
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007fb2194f2
        /lib64/ld-linux-x86-64.so.2 (0x00007fb219605000)
        libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007
```

Qt C++ GUI application

```
mwales@Metroid:~/checkouts/WildcatPracticeCtf/year_2023_2024/ctf3/39_meme_viewer_2/build$ ldd ./meme_viewer_v2
        linux-vdso.so.1 (0x00007ffe9ccd3000)
        libQt5Widgets.so.5 => /lib/x86_64-linux-gnu/libQt5Widgets.so.5 (0x00007f76d4000000)
        libQt5Gui.so.5 => /lib/x86_64-linux-gnu/libQt5Gui.so.5 (0x00007f76d3800000)
        libQt5Core.so.5 => /lib/x86_64-linux-gnu/libQt5Core.so.5 (0x00007f76d3200000)
        libstdc++.so.6 => /lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f76d2e00000)
        libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f76d47e0000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f76d2a00000)
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f76d46f7000)
        libGL.so.1 => /lib/x86_64-linux-gnu/libGL.so.1 (0x00007f76d3f79000)
        libpng16.so.16 => /lib/x86_64-linux-gnu/libpng16.so.16 (0x00007f76d3f3e000)
        libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f76d46db000)
        libharfbuzz.so.0 => /lib/x86_64-linux-gnu/libharfbuzz.so.0 (0x00007f76d3131000)
        libmd4c.so.0 => /lib/x86_64-linux-gnu/libmd4c.so.0 (0x00007f76d46c7000)
        libdouble-conversion.so.3 => /lib/x86_64-linux-gnu/libdouble-conversion.so.3 (0x00007f76d3f29000)
        libicui18n.so.70 => /lib/x86_64-linux-gnu/libicui18n.so.70 (0x00007f76d2600000)
        libicuuc.so.70 => /lib/x86_64-linux-gnu/libicuuc.so.70 (0x00007f76d2405000)
        libpcre2-16.so.0 => /lib/x86_64-linux-gnu/libpcre2-16.so.0 (0x00007f76d3776000)
        libzstd.so.1 => /lib/x86_64-linux-gnu/libzstd.so.1 (0x00007f76d3062000)
        libglib-2.0.so.0 => /lib/x86_64-linux-gnu/libglib-2.0.so.0 (0x00007f76d2cc6000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f76d4923000)
        libGLdispatch.so.0 => /lib/x86_64-linux-gnu/libGLdispatch.so.0 (0x00007f76d2948000)
        libGLX.so.0 => /lib/x86_64-linux-gnu/libGLX.so.0 (0x00007f76d3ef5000)
        libfreetype.so.6 => /lib/x86_64-linux-gnu/libfreetype.so.6 (0x00007f76d233d000)
        libgraphite2.so.3 => /lib/x86_64-linux-gnu/libgraphite2.so.3 (0x00007f76d303b000)
        libicudata.so.70 => /lib/x86_64-linux-gnu/libicudata.so.70 (0x00007f76d0600000)
        libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007f76d2c50000)
        libX11.so.6 => /lib/x86_64-linux-gnu/libX11.so.6 (0x00007f76d04c0000)
        libbrotlidec.so.1 => /lib/x86_64-linux-gnu/libbrotlidec.so.1 (0x00007f76d3ee7000)
        libxcb.so.1 => /lib/x86_64-linux-gnu/libxcb.so.1 (0x00007f76d2313000)
        libbrotlicommon.so.1 => /lib/x86_64-linux-gnu/libbrotlicommon.so.1 (0x00007f76d2c2d000)
        libXau.so.6 => /lib/x86_64-linux-gnu/libXau.so.6 (0x00007f76d3ee1000)
        libXdmcp.so.6 => /lib/x86_64-linux-gnu/libXdmcp.so.6 (0x00007f76d376e000)
        libbsd.so.0 => /lib/x86_64-linux-gnu/libbsd.so.0 (0x00007f76d2930000)
        libmd.so.0 => /lib/x86_64-linux-gnu/libmd.so.0 (0x00007f76d3761000)
```

Dynamically Linked Function in Binja

- White = built-in functions
- Green = current function your looking at
- Orange = functions from library / dynamically linked / weak symbols
    - Calls to these functions use GOT / PLT to look up and load libraries ***when needed***

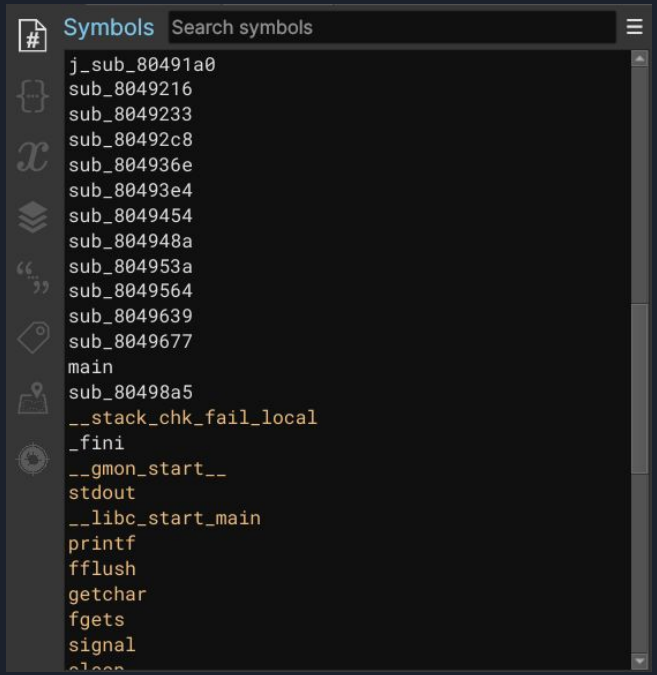



```
0x804c004 .got.plt (PROGBITS) {0x804c000-0x804c03c} Writable data

0804c004  int32_t data_804c004 = 0x0
0804c008  int32_t data_804c008 = 0x0
0804c00c  void (* const __libc_start_main)(int32_t (* main)(int32_t argc, char** argv, ch
0804c010  int32_t (* const printf)(char const* format, ...) = printf
0804c014  int32_t (* const fflush)(FILE* fp) = fflush
0804c018  int32_t (* const getchar)() = getchar
0804c01c  char* (* const fgets)(char* buf, int32_t n, FILE* fp) = fgets
0804c020  __sighandler_t (* const signal)(int32_t sig, __sighandler_t handler) = signal
0804c024  uint32_t (* const sleep)(uint32_t seconds) = sleep
0804c028  void (* const __stack_chk_fail)() __noreturn = __stack_chk_fail
0804c02c  int32_t (* const puts)(char const* str) = puts
0804c030  void (* const exit)(int32_t status) __noreturn = exit
0804c034  FILE* (* const fopen)(char const* filename, char const* mode) = fopen
0804c038  int32_t (* const putchar)(int32_t c) = putchar
```

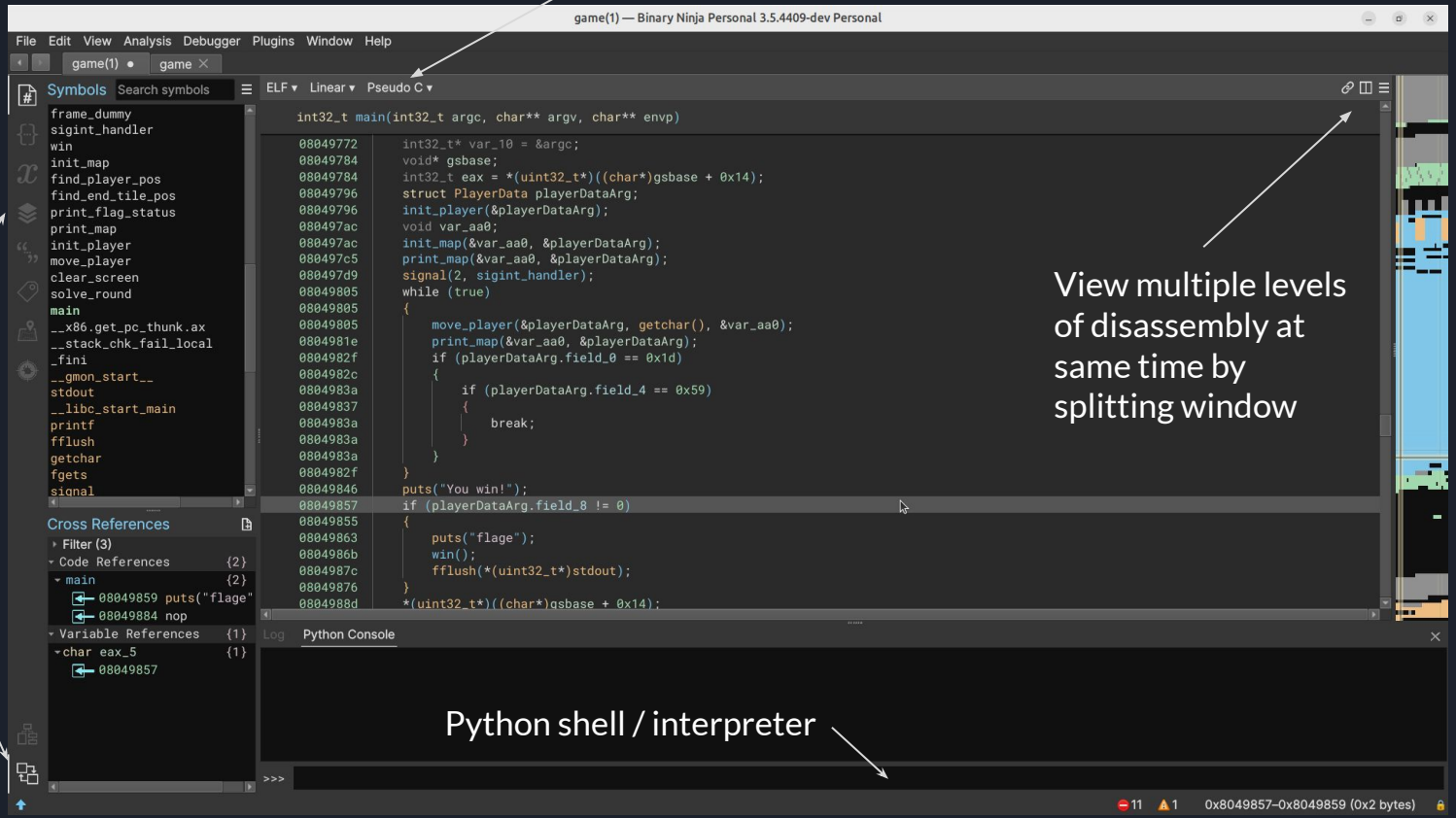
Stripped Binaries

- Notice how the built-in functions had names?
- GNU Strip will remove all symbol names from an ELF
  - `strip ./game`
- sub_address: functions that probably had name before
  - We don't even know these are really functions, Binary Ninja via analysis thinks these are functions (it's pretty good at figuring them out)
- Dynamically linked functions still have to be looked up by name
- Statically linked binaries make all library functions built-in functions
  - Makes binary much larger
  - More portable / less dependencies
  - Makes reverse engineer's life miserable



Binary Ninja GUI

Disassembly Level (very helpful!)



Buttons to change side dock contents

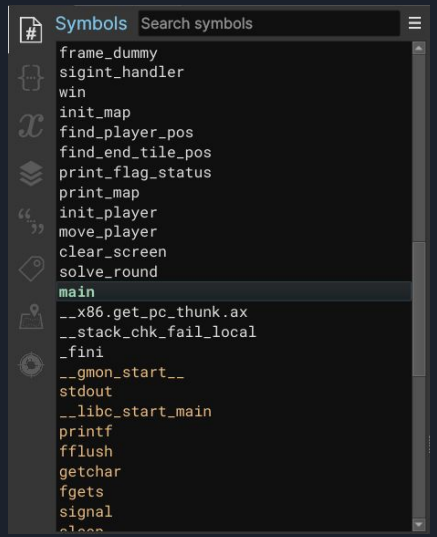
View multiple levels of disassembly at same time by splitting window

Python shell / interpreter

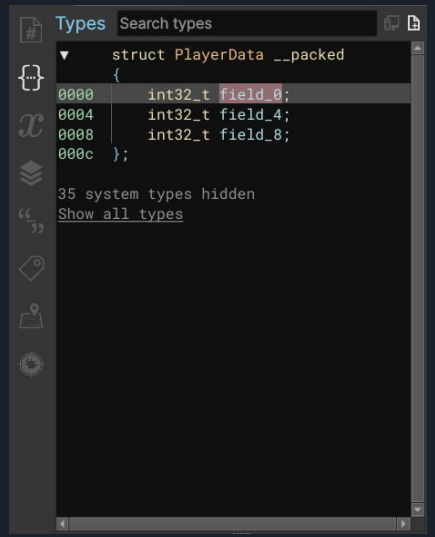
Side Dock Modes

Symbols:
- Functions
- Global Variables

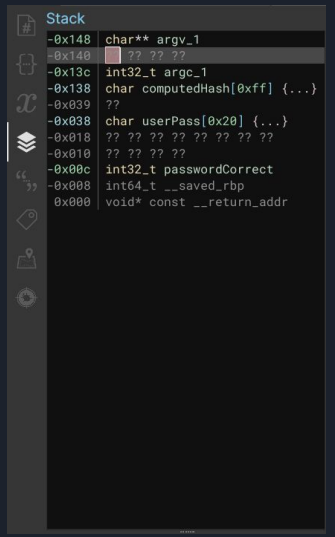
You can filter / search list



Structures / Types



Stack / Local Vars
- Visualize where variables are / how many bytes
- What buffer overflows effect



What are we trying to do?

- Identify structures (usually need to figure out the size of the structure)
  - Things that look like offsets from pointer / arrays
  - Group of related variables passed by pointer to functions
- Name functions, variables, structure fields
  - Guess what the variables should be called based on how code uses them
  - Shortcut key n : rename
- Set variable types
  - Shortcut key y : set type
  - Shortcut key d : rotate integer types
  - Shortcut key u : undefine
- Repeat
  - Shortcut key ESC : go back to previous location

Trying to get Binary Ninja disassembly / decompilation to look more similar to what the original source would look like so we can easily understand it

Links

- https://event.fit.edu/gencyber/
-