Finding Xori

Malware Analysis Triage with Automated Disassembly

Amanda Rousseau
Rich Seymour

About Us

Project started in Jan 2018

- Approx 6 months
- ~36K lines of code

Amanda Rousseau

Sr. Malware Researcher,

Endgame, Inc.

@malwareunicorn

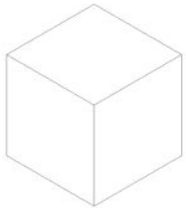
Rich Seymour

Sr. Data Scientist,

Endgame, Inc.

@rseymour

Quick Overview



The Current State of Disassemblers

Brief overview of pros and cons with current popular open source PE disassemblers.

Functionality & Features

Overview how we pulled together the different aspects of disassemblers and emulator

Usage & Demo

How the output is used for automation. Applying the tool on various malware samples and shellcode.

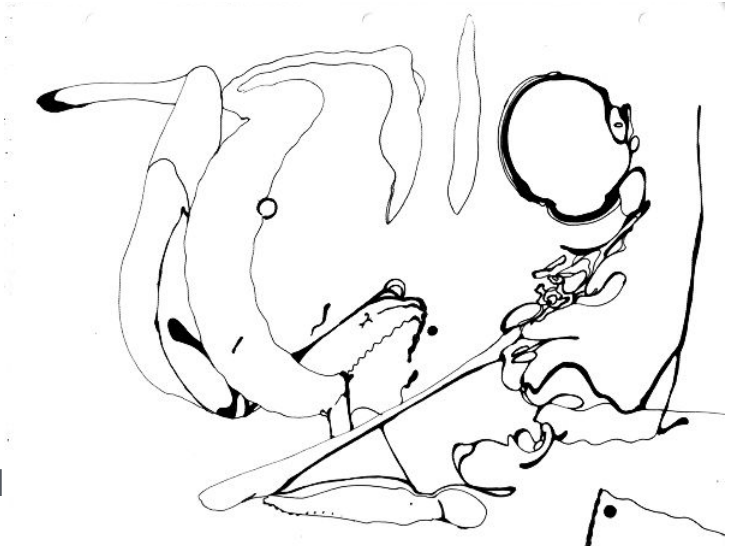
The Problem

There are millions of malware samples to look at and a few reverse engineers.

We need to change the way we are going about this if we are going to keep up.

How to leverage large scale disassembly in an automated way with many samples?

- Improve the scalability in malware analysis
- Integration and automation



Present Day Common Disassemblers

	Capstone	Radare2	IDA Pro	Hopper	Binary Ninja
Size	small	small	large	medium	large
Stability	✔	✖	✔	✔	✔
Price	-	-	$$$	$	$$
Cross Platform	✔	~	✔	✖	✔
Usability	~	~	✔	~	~
Accuracy	~	~	✔	~	~
Integration	✔	~	✖	✖	✖

Requirements

- Fast development
- Stability and resilience
- Cross platform
- Output can be easily integrated
- Ease of use
- Core feature set
- Output accuracy

Evaluating Disassemblers

The first step - Diving into the code:

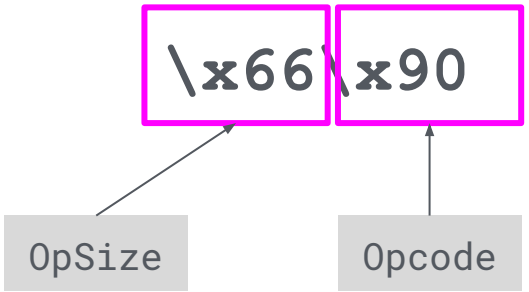
- Verifying the accuracy of various disassemblers
- Understand each of their strengths and limitations

We adopted different aspects of disassemblers and emulator modules.

- Much of Capstone is also based on the LLVM & GDB repositories
- QEMU is the emulation is straightforward, easy to understand
- Converted some of the logic into Rust, while also fixing a few bugs along the way.

Evaluating Example

X86 32bit:



XCHG AX, AX [Objdump] ✔

XCHG AX, AX [IDA Pro] ✔

NOP [Capstone] ✖

NOP [Distorm] ✖

Developed in Rust

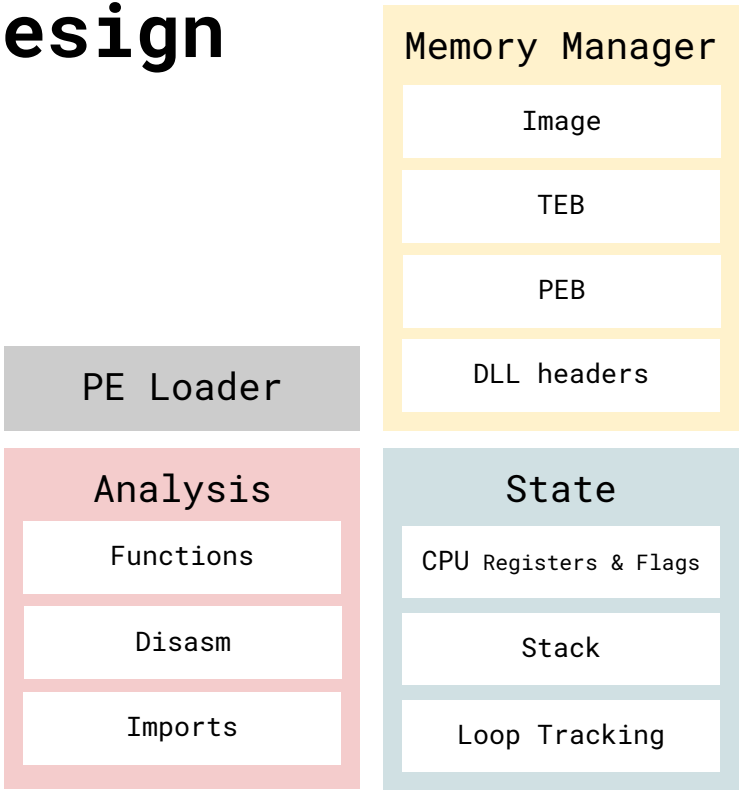
Why Rust?

- Same capabilities in C\C++
- Stack protection
- Proper memory handling (guaranteed memory safety)
- Provides stability and speed (minimal runtime)
- Faster development
- Helpful compiler

Current Features

- Open source
- Supports i386 and x86-64 architecture only at the moment
- Displays strings based on referenced memory locations
- Manages memory
- Outputs Json
- 2 modes: with or without emulation
  - Light Emulation - meant to enumerate all paths (Registers, Stack, Some Instructions)
  - Full Emulation - only follows the code’s path (Slow performance)
- Simulated TEB & PEB structures
- Evaluates functions based on DLL exports

Design



PE Loader

Handles the loading of the PE image into memory and sets up the TEB/PEB as well as initializing the offsets to loaded DLLs and import table.

Memory Manager

This structure contains all of the mmap memory for the Image, TEB/PEB, and DLL headers. Accessors for Read & Write to avoid errors in inaccessible memory.

Analysis

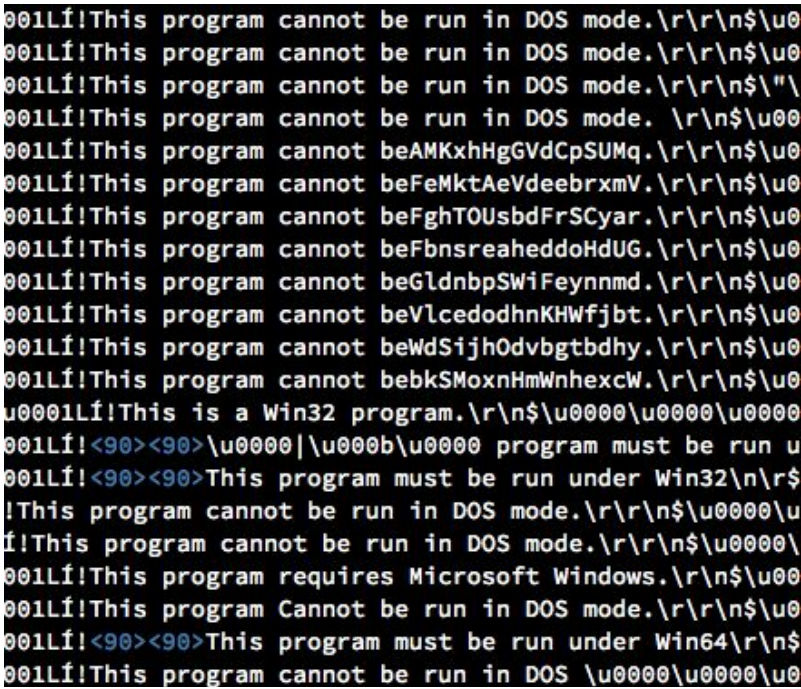
The core container for the disassembly, functions, and imports.

State

This structure contains the CPU state of the registers & flags, a new copy of the stack, and short circuiting for looping during emulation.

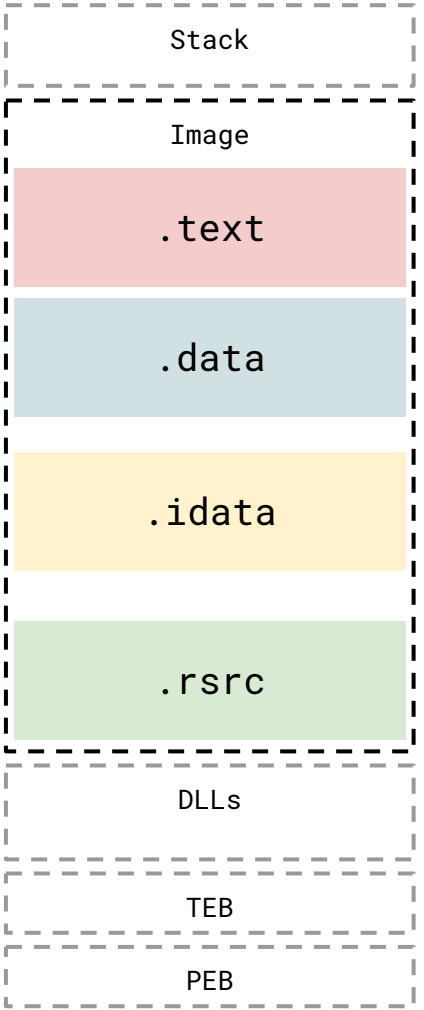
Roll your own PE Parser

- Although a few Rust PE parsers exist: goblin, pe-rs we decided to create our own.
- Chose to write it using the **nom** parser combinator framework
- Ideally less error prone due to safe macro constructions
- Many lessons learned
- From a historical perspective a PE parser start reading a 16 bit DOS file
- Then optionally switches to a PE32 or a PE32+
- This is like a history of DOS and Microsoft Windows in a single parser.



Analysis Enrichment

- The header is used to build the memory sections of the PE Image
- Similar to the PE loader in windows, it will load the image similar to how it would be loaded in the addressable memory. Where the imports are given memory address, rewritten in the image.



Symbols

- We needed a way to load DLL exports and header information without doing it natively.
- Built a parser that would generate json files for consumption called pesymbols.
- Instead of relying on the Import Table of the PE, it generates virtual addresses of the DLL and API in the Image's Import Table. This way you can track the actual address of the function being pushed into various registers.
- The virtual address start is configurable as well as the json location.

Configurable in xori.json

```
"dll_address32": 1691680768, 0x64D50000
"dll_address64": 8789194768384, 0x7FE64D50000
"function_symbol32":
"./src/analysis/symbols/generated_user_syswow64.json",
"function_symbol64":
"./src/analysis/symbols/generated_user_system32.json",
...
```

Generated_user_syswow64.json

```
{
    "name": "kernel32.dll",
    "exports": [
      {
        "address": 696814,
        "name": "AcquireSRWLockExclusive",
        "ordinal": 1,
        "forwarder": true,
        "forwarder_name": "NTDLL.RtlAcquireSRWLockExclusive"
      },
      {
        "address": 696847,
        "name": "AcquireSRWLockShared",
        "ordinal": 2,
        "forwarder": true,
        "forwarder_name": "NTDLL.RtlAcquireSRWLockShared"
      },
      ...
```

Dealing with Dynamic API Calls

TEB/PEB

The TEB and PEB structures are simulated based on the the imports and known dlls in a windows 7 environment.

Memory Management

Segregated memory for the local memory storage such as the stack.

The Stack

If references to functions are pushed into a register or stack will be able to be tracked.

Dealing with Dynamic API Calls

Header Imports
"ExitProcess"
"GetLastError"
"GetLocalTime"
"GetModuleHandleA"

Dynamic Imports
"LoadLibraryA"
"VirtualProtect"
"ShellExecuteA"

```
                                00          mov [0x401000], eax
                                00          push 0x401041 ; LoadLibraryA
                                40 00       push [0x401000]
                                00          call 0x4012cb
                                            cmp eax, 0x0
                                00 00       je 0x4013da
                                00          mov [0x401004], eax ; wI
                                00          push 0x40104e ; VirtualProtect
0x401115        FF 35 00 10 40 00           push [0x401000]
0x40111b        E8 AB 01 00 00              call 0x4012cb
0x401120        83 F8 00                    cmp eax, 0x0
0x401123        0F 84 B1 02 00 00           je 0x4013da
0x401129        A3 08 10 40 00              mov [0x401008], eax
0x40112e        6A 00                       push 0x0
                                            push 0x0
                                00          push 0x40101c ; shell32.dll
                                40 00       call [0x401004] ; kernel32.dll!LoadLibraryA
                                00          mov [0x40100c], eax
                                00          push 0x401033 ; ShellExecuteA
                                40 00       push [0x40100c]
                                00          call 0x4012cb
0x401152        A3 10 10 40 00              mov [0x401010], eax
```

Loads LoadLibrary from the PEB

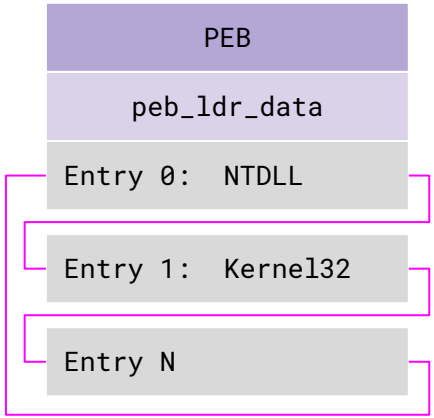
Stores the address into ptr [0x401004]

Calls the new ptr

TEB & PEB

In Rust, you can serialize structs into vectors of bytes. This way you can allow the assembly emulation to access them natively while also managing the access.

```
let teb_binary: Vec<u8> =
serialize(&teb_struct).unwrap();
```




```
#[derive(Serialize, Deserialize)]
struct ThreadInformationBlock32
{
    // reference: https://en.wikipedia.org/wiki/Win32_Thread_Information_Block
    seh_frame:              u32,  //0x00
    stack_base:             u32,  //0x04
    stack_limit:            u32,  //0x08
    subsystem_tib:          u32,  //0x0C
    fiber_data:             u32,  //0x10
    arbitrary_data:         u32,  //0x14
    self_addr:              u32,  //0x18
    //End                   of NT subsystem independent part
    environment_ptr:        u32,  //0x1C
    process_id:             u32,  //0x20
    thread_id:              u32,  //0x24
    active_rpc_handle:      u32,  //0x28
    tls_addr:               u32,  //0x2C
    peb_addr:               u32,  //0x30
    last_error:             u32,  //0x34
    critical_section_count: u32,  //0x38
    csr_client_thread:      u32,  //0x3C
    win32_thread_info:      u32,  //0x40
    win32_client_info:      [u32; 31],    //0x44
    ...
```

Code vs. Data

- padding after a non-returning call
- calls misidentified as non-returning
- repetitive instruction sequences
- missed computed jump targets
- false computed jump targets
- missed opcode prefixes
- code following unconditional branches
- code following returns
- code following conditional branches

Padding after a non-returning call

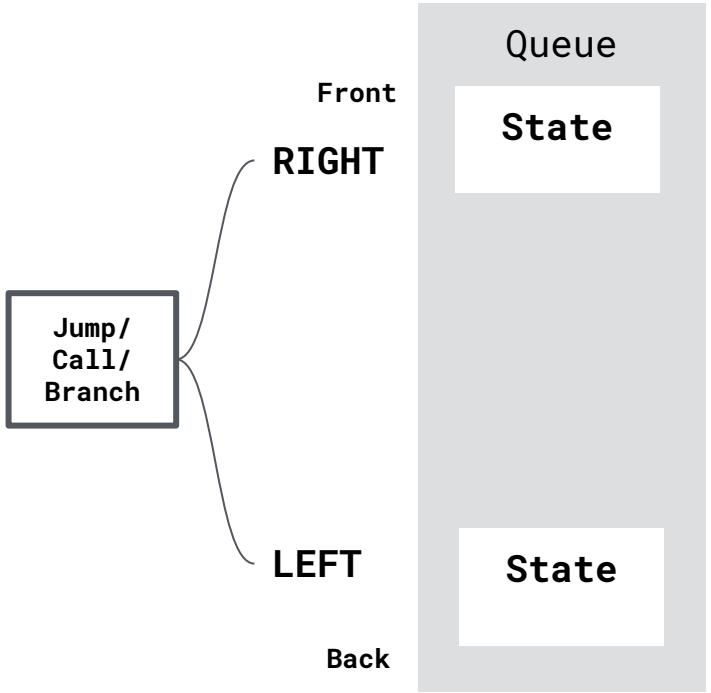
```
0x40116c 68 B0 12 40 00        push 0x4012b0 ; VB5!
0x401171 E8 F0 FF FF FF        call 0x401166 ; FUNC 0x40116c END
0x401178 00 00 00 00           db 0x0
0x40117c 30 00 00 00           db 0x30
0x401180 40 00 00 00           db 0x40
0x401184 00 00 00 00           db 0x0
```

Padding after returns

```
0x14000286a 39 05 C0 27 00 00     cmp [rip+0x27c0], eax
0x140002870 0F 95 C0              setne al
0x140002873 C3                    ret  ; FUNC 0x140002868 END
0x140002878 CC CC CC CC CC CC CC CC  db 0xcccccccccccccccc
0x140002880 FF 25 A2 08 00 00     jmp [rip+0x8a2]
```

Handling Branches & Calls

- Branches and calls have 2 directions
  - Left & Right
- In light emulation mode, both the left and right directions are followed
- Each direction is placed onto a queue with it’s own copy of the state.
- Any assembly not traversed will not be analyzed.
- All function calls are tracked for local and import table mapping.



Handling Looping

- Infinite loops are hard to avoid
- Built a way to configure the maximum amount of loops a one can take
  - Forward
  - Backward
  - Standard Loop
- The state contains the looping information
- Once the maximum is reached, it will disable the loop

Configurable in xori.json

```
"loop_default_case": 4000,
...
```

Signature Analysis

- We needed a way to add signatures fast
- When bytes are analyzed as data, you can apply signatures (i.e Function Headers)
- Importing FLIRT Signatures

```
Signature{
  name: "_standard_func_header",
  pattern: &[
    r"\x55\x8B\xEC", //push ebp, mov ebp,esp
    r"\x55\x89\xE5", //push ebp, mov ebp,esp
   ],
}
```

```
<leading bytes> <CRC16 len> <CRC16> <total length> <public name(s)> <referenced name(s)>

558BEC56FC8B750C8B4E0833CEE8.....6A0056FF7614FF760C6A00FF7510 07
5FB2 0031 :0000 __CatchGuardHandler ^000E @__security_check_cookie@4
^0027 ___InternalCxxFrameHandler

558BEC8B4D0C568B7508890EE8.....8B4824894E04E8.....8970248B 04
7D88 0024 :0000 __CreateFrameInfo ^000D ___vcrt_getptd
```

Automation for Bulk Analysis

- On the Ember Test set of 1000 files, Xori processes a sample in 1.25 seconds. With 8 cores, 1000 files takes about 20 min
- Creates *valid* JSON output of PE features, control flow graph, functions from binary files allowing bulk data analysis: clustering, outlier detection and visualization.
- You can then easily throw Xori output into a database, document store or do a little data science at the command line

```
$ jq '.import_table?|map(.import_address_list)?|map(.[].func_name)?|.[]' *header.json |sort | uniq -c
|sort -nr|grep -i crypt
     32 "CryptReleaseContext"
     23 "CryptGenRandom"
     19 "CryptAcquireContextA"
     12 "CryptAcquireContextW"
      7 "CryptHashData"
      7 "CryptGetHashParam"
      7 "CryptDestroyHash"
      ...
```

Examples

Simplest Way to Run Xori

```
Cd ./xori

Cargo build --release

./target/release/xori -f wanacry.exe
```

Basic Disassembler

```
extern crate xori;
use std::fmt::Write;
use xori::disasm::*;
use xori::arch::x86::archx86::X86Detail;

fn main()
{
    let xi = Xori { arch: Arch::ArchX86, mode: Mode::Mode32 };
    let start_address = 0x1000;
    let binary32 = b"\xe9\x1e\x00\x00\x00\xb8\x04\
    \x00\x00\x00\xbb\x01\x00\x00\x00\x59\xba\x0f\
    \x00\x00\x00\xcd\x80\xb8\x01\x00\x00\x00\xbb\
    \x00\x00\x00\x00\xcd\x80\xe8\xdd\xff\xff\xff\
    \x48\x65\x6c\x6c\x6f\x2c\x20\x57\x6f\x72\x6c\
    \x64\x21\x0d\x0a";

    let mut vec: Vec<Instruction<X86Detail>> = Vec::new();
    xi.disasm(binary32, binary32.len(),
        start_address, start_address, 0, &mut vec);
    if vec.len() > 0
    {
        //Display values
        for instr in vec.iter_mut()
        {
            let addr: String = format!("0x{:x}", instr.address);
            println!("{:16} {:20} {} {}", addr,
                hex_array(&instr.bytes, instr.size),
                instr.mnemonic, instr.op_str);
        }
    }
}
```

```
xori $RUST_BACKTRACE=1 cargo run --release --example simple_disasmx86
   Compiling xori v0.0.1 (file:///Users/amanda/Documents/Projects/xori)
    Finished release [optimized + debuginfo] target(s) in 1.13s
     Running `target/release/examples/simple_disasmx86`
0x1000           E9 1E 00 00 00       jmp 0x1023
0x1005           B8 04 00 00 00       mov eax, 0x4
0x100a           BB 01 00 00 00       mov ebx, 0x1
0x100f           59                   pop ecx
0x1010           BA 0F 00 00 00       mov edx, 0xf
0x1015           CD 80                int 0x80
0x1017           B8 01 00 00 00       mov eax, 0x1
0x101c           BB 00 00 00 00       mov ebx, 0x0
0x1021           CD 80                int 0x80
0x1023           E8 DD FF FF FF       call 0x1005
0x1028           48                   dec eax
0x1029           65 6C                insb es:[edi], dx
0x102b           6C                   insb es:[edi], dx
0x102c           6F                   outsd dx, [esi]
0x102d           2C 20                sub al, 0x20
0x102f           57                   push edi
0x1030           6F                   outsd dx, [esi]
0x1031           72 6C                jb 0x109f
0x1033           64                   db 0x64
0x1034           21                   db 0x21
0x1035           0D                   db 0xd
0x1036           0A                   db 0xa
```

Binary File Disassembler

```
extern crate xori;
extern crate serde_json;
use serde_json::Value;
use std::path::Path;
use xori::analysis::analyze::analyze;
use xori::disasm::*;

fn main()
{
    let mut binary32 = b"\xe9\x1e\x00\x00\x00\xb8\x04\
    \x00\x00\x00\xbb\x01\x00\x00\x00\x59\xba\x0f\
    \x00\x00\x00\xcd\x80\xb8\x01\x00\x00\x00\xbb\
    \x00\x00\x00\x00\xcd\x80\xe8\xdd\xff\xff\xff\
    \x48\x65\x6c\x6c\x6f\x2c\x20\x57\x6f\x72\x6c\
    \x64\x21\x0d\x0a".to_vec();

    let mut config_map: Option<Value> = None;
    if Path::new("xori.json").exists()
    {
        config_map = read_config(&Path::new("xori.json"));
    }
    match analyze(&Arch::ArchX86, &mut binary32, &config_map)
    {
        Some(analysis)=>{
            if !analysis.disasm.is_empty(){
                println!("{}", analysis.disasm);
            }
        },
        None=>{},
    }
}
```

```
xori $RUST_BACKTRACE=1 cargo run --release --example simple_bin
   Compiling xori v0.0.1 (file:///Users/amanda/Documents/Projects/xori)
    Finished release [optimized + debuginfo] target(s) in 43.25s
     Running `target/release/examples/simple_bin`
0x1000          E9 1E 00 00 00      jmp 0x1023
0x1005          B8 04 00 00 00      mov eax, 0x4
0x100a          BB 01 00 00 00      mov ebx, 0x1
0x100f          59                  pop ecx ; Hello, World!
0x1010          BA 0F 00 00 00      mov edx, 0xf
0x1015          CD 80               int 0x80
0x1017          B8 01 00 00 00      mov eax, 0x1
0x101c          BB 00 00 00 00      mov ebx, 0x0
0x1021          CD 80               int 0x80 ; FUNC 0x1005 END
0x1023          E8 DD FF FF FF      call 0x1005
0x1028          48                  dec eax
0x1029          65 6C               insb es:[edi], dx
0x102b          6C                  insb es:[edi], dx
0x102c          6F                  outsd dx, [esi]
0x102d          2C 20               sub al, 0x20
0x102f          57                  push edi
0x1030          6F                  outsd dx, [esi]
0x1031          72 6C               jb 0x109f
0x1033          64                  db 0x64
0x1034          21                  db 0x21
0x1035          0D                  db 0xd
0x1036          0A                  db 0xa
```

WanaCry Ransomware

Xori

```
0x408140        83 EC 50            sub esp, 0x50
0x408143        56                  push esi
0x408144        57                  push edi
0x408145        B9 0E 00 00 00      mov ecx, 0xe
0x40814a        BE D0 13 43 00      mov esi, 0x4313d0 ; http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
0x40814f        8D 7C 24 08         lea edi, [esp+0x8]
0x408153        33 C0               xor eax, eax
0x408155        F3 A5               rep movsd [esi], es:[edi]
0x408157        A4                  movsb [esi], es:[edi]
0x408158        89 44 24 41         mov [esp+0x41], eax
0x40815c        89 44 24 45         mov [esp+0x45], eax
0x408160        89 44 24 49         mov [esp+0x49], eax
0x408164        89 44 24 4D         mov [esp+0x4d], eax
0x408168        89 44 24 51         mov [esp+0x51], eax
0x40816c        66 89 44 24 55      mov [esp+0x55], ax
0x408171        50                  push eax
0x408172        50                  push eax
0x408173        50                  push eax
0x408174        6A 01               push 0x1
0x408176        50                  push eax
0x408177        88 44 24 6B         mov [esp+0x6b], al
0x40817b        FF 15 34 A1 40 00   call [0x40a134] ; wininet.dll!InternetOpenA
0x408181        6A 00               push 0x0
0x408183        68 00 00 00 84      push 0x84000000
0x408188        6A 00               push 0x0
0x40818a        8D 4C 24 14         lea ecx, [esp+0x14]
0x40818e        8B F0               mov esi, eax
0x408190        6A 00               push 0x0
0x408192        51                  push ecx
0x408193        56                  push esi
0x408194        FF 15 38 A1 40 00   call [0x40a138] ; wininet.dll!InternetOpenUrlA
0x40819a        8B F8               mov edi, eax
0x40819c        56                  push esi
0x40819d        8B 35 3C A1 40 00   mov esi, [0x40a13c] ; p\xFB&e
0x4081a3        85 FF               test edi, edi
0x4081a5        75 15               jne 0x4081bc
0x4081a7        FF D6               call esi ; wininet.dll!InternetCloseHandle
0x4081a9        6A 00               push 0x0
0x4081ab        FF D6               call esi ; wininet.dll!InternetCloseHandle
0x4081ad        E8 DE FE FF FF      call 0x408090
0x4081b2        5F                  pop edi ; _3
0x4081b3        33 C0               xor eax, eax
0x4081b5        5E                  pop esi
0x4081b6        83 C4 50            add esp, 0x50
0x4081b9        C2 10 00            ret 0x10 ; FUNC 0x408140 END
0x4081bc        FF D6               call esi ; wininet.dll!InternetCloseHandle
0x4081be        57                  push edi
0x4081bf        FF D6               call esi ; wininet.dll!InternetCloseHandle
0x4081c1        5F                  pop edi
0x4081c2        33 C0               xor eax, eax
0x4081c4        5E                  pop esi
0x4081c5        83 C4 50            add esp, 0x50
0x4081c8        C2 10 00            ret 0x10 ; FUNC 0x408140 END
```

IDA Pro

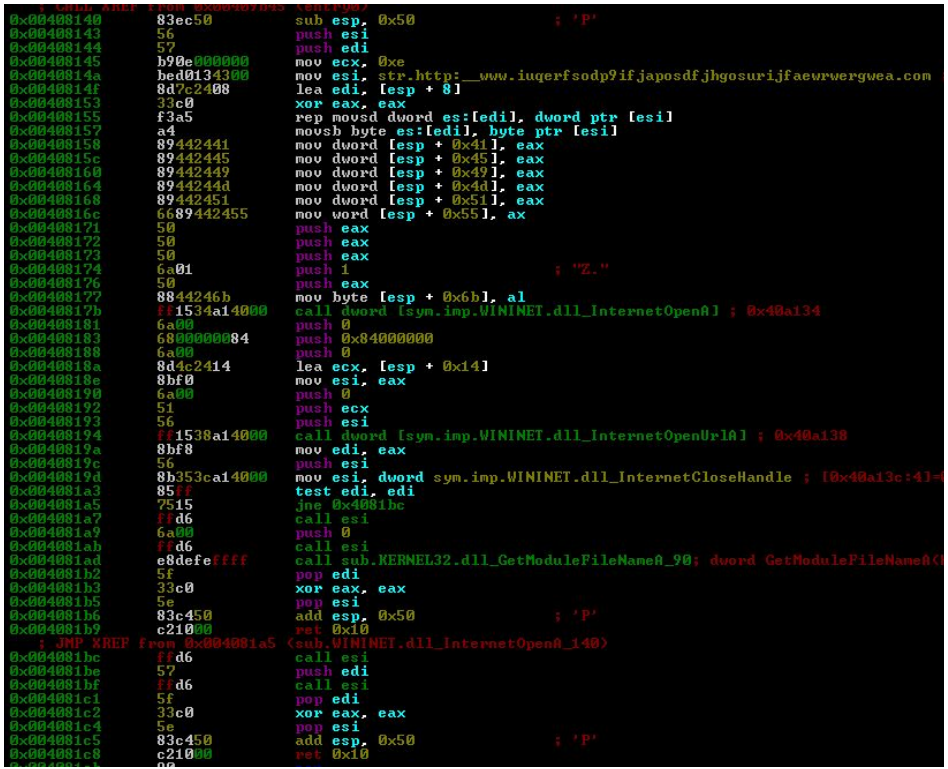
```
.text:00408140              sub     esp, 50h
.text:00408143              push    esi
.text:00408144              push    edi
.text:00408145              mov     ecx, 0Eh
.text:0040814A              mov     esi, offset aHttpWww_iuqerf ; "http://www.iuqerfsodp9ifjaposdfjhgosuri"
.text:0040814F              lea     edi, [esp+58h+szUrl]
.text:00408153              xor     eax, eax
.text:00408155              rep movsd
.text:00408157              movsb
.text:00408158              mov     [esp+58h+var_17], eax
.text:0040815C              mov     [esp+58h+var_13], eax
.text:00408160              mov     [esp+58h+var_F], eax
.text:00408164              mov     [esp+58h+var_B], eax
.text:00408168              mov     [esp+58h+var_7], eax
.text:0040816C              mov     [esp+58h+var_3], ax
.text:00408171              push    eax             ; dwFlags
.text:00408172              push    eax             ; lpszProxyBypass
.text:00408173              push    eax             ; lpszProxy
.text:00408174              push    1               ; dwAccessType
.text:00408176              push    eax             ; lpszAgent
.text:00408177              mov     [esp+6Ch+var_1], al
.text:0040817B              call    ds:InternetOpenA
.text:00408181              push    0               ; dwContext
.text:00408183              push    84000000h       ; dwFlags
.text:00408188              push    0               ; dwHeadersLength
.text:0040818A              lea     ecx, [esp+64h+szUrl]
.text:0040818E              mov     esi, eax
.text:00408190              push    0               ; lpszHeaders
.text:00408192              push    ecx             ; lpszUrl
.text:00408193              push    esi             ; hInternet
.text:00408194              call    ds:InternetOpenUrlA
.text:0040819A              mov     edi, eax
.text:0040819C              push    esi             ; hInternet
.text:0040819D              mov     esi, ds:InternetCloseHandle
.text:004081A3              test    edi, edi
.text:004081A5              jnz     short loc_4081BC
.text:004081A7              call    esi ; InternetCloseHandle
.text:004081A9              push    0               ; hInternet
.text:004081AB              call    esi ; InternetCloseHandle
.text:004081AD              call    sub_408090
.text:004081B2              pop     edi
.text:004081B3              xor     eax, eax
.text:004081B5              pop     esi
.text:004081B6              add     esp, 50h
.text:004081B9              retn    10h
```

WanaCry Ransomware

Xori

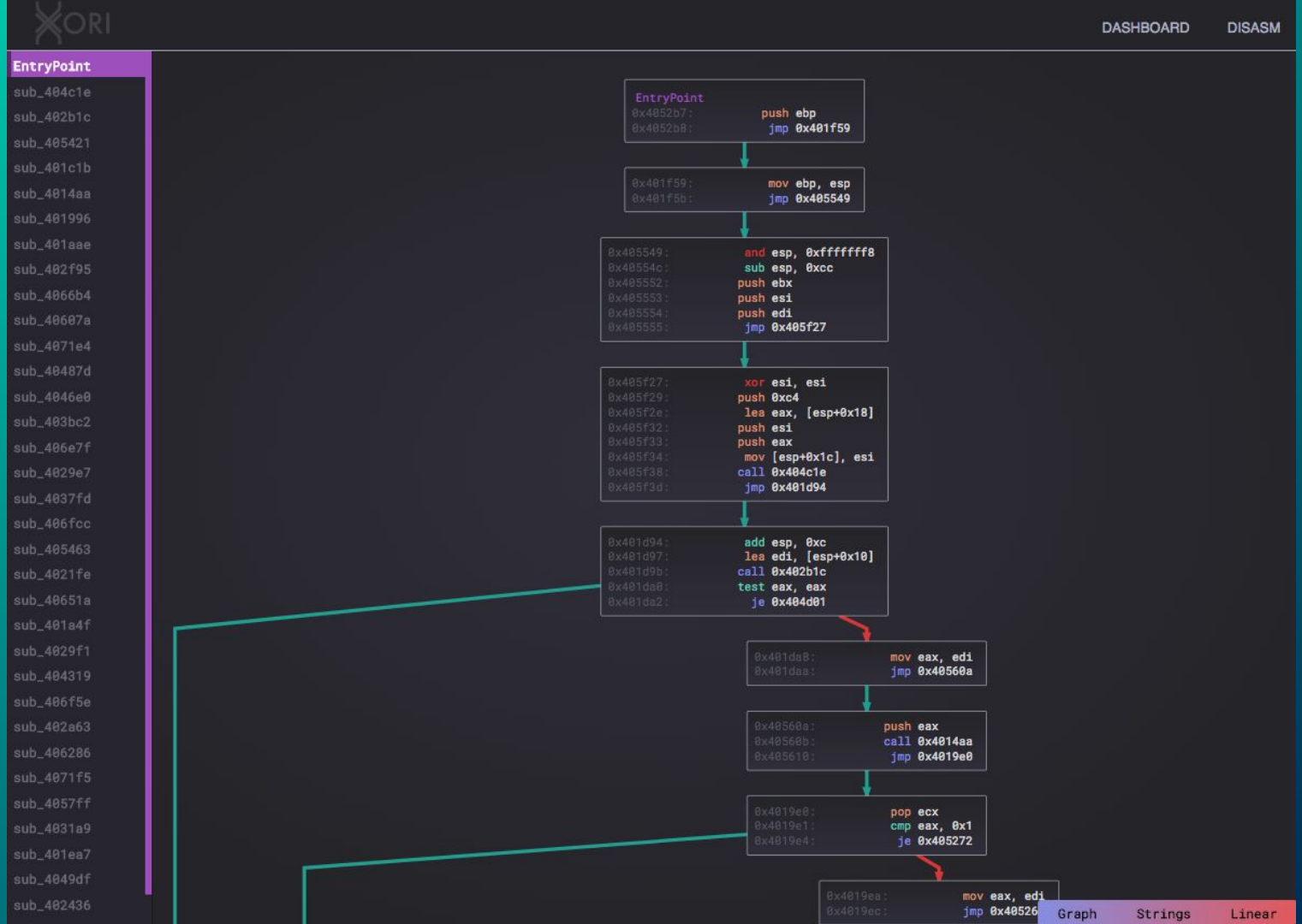
```
0x408140        83 EC 50            sub esp, 0x50
0x408143        56                  push esi
0x408144        57                  push edi
0x408145        B9 0E 00 00 00      mov ecx, 0xe
0x40814a        BE D0 13 43 00      mov esi, 0x4313d0 ; http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
0x40814f        8D 7C 24 08         lea edi, [esp+0x8]
0x408153        33 C0               xor eax, eax
0x408155        F3 A5               rep movsd [esi], es:[edi]
0x408157        A4                  movsb [esi], es:[edi]
0x408158        89 44 24 41         mov [esp+0x41], eax
0x40815c        89 44 24 45         mov [esp+0x45], eax
0x408160        89 44 24 49         mov [esp+0x49], eax
0x408164        89 44 24 4D         mov [esp+0x4d], eax
0x408168        89 44 24 51         mov [esp+0x51], eax
0x40816c        66 89 44 24 55      mov [esp+0x55], ax
0x408171        50                  push eax
0x408172        50                  push eax
0x408173        50                  push eax
0x408174        6A 01               push 0x1
0x408176        50                  push eax
0x408177        88 44 24 6B         mov [esp+0x6b], al
0x40817b        FF 15 34 A1 40 00   call [0x40a134] ; wininet.dll!InternetOpenA
0x408181        6A 00               push 0x0
0x408183        68 00 00 00 84      push 0x84000000
0x408188        6A 00               push 0x0
0x40818a        8D 4C 24 14         lea ecx, [esp+0x14]
0x40818e        8B F0               mov esi, eax
0x408190        6A 00               push 0x0
0x408192        51                  push ecx
0x408193        56                  push esi
0x408194        FF 15 38 A1 40 00   call [0x40a138] ; wininet.dll!InternetOpenUrlA
0x40819a        8B F8               mov edi, eax
0x40819c        56                  push esi
0x40819d        8B 35 3C A1 40 00   mov esi, [0x40a13c] ; p\xFB&e
0x4081a3        85 FF               test edi, edi
0x4081a5        75 15               jne 0x4081bc
0x4081a7        FF D6               call esi ; wininet.dll!InternetCloseHandle
0x4081a9        6A 00               push 0x0
0x4081ab        FF D6               call esi ; wininet.dll!InternetCloseHandle
0x4081ad        E8 DE FE FF FF      call 0x408090
0x4081b2        5F                  pop edi ; _3
0x4081b3        33 C0               xor eax, eax
0x4081b5        5E                  pop esi
0x4081b6        83 C4 50            add esp, 0x50
0x4081b9        C2 10 00            ret 0x10 ; FUNC 0x408140 END
0x4081bc        FF D6               call esi ; wininet.dll!InternetCloseHandle
0x4081be        57                  push edi
0x4081bf        FF D6               call esi ; wininet.dll!InternetCloseHandle
0x4081c1        5F                  pop edi
0x4081c2        33 C0               xor eax, eax
0x4081c4        5E                  pop esi
0x4081c5        83 C4 50            add esp, 0x50
0x4081c8        C2 10 00            ret 0x10 ; FUNC 0x408140 END
```

Radare2



```
0x00408140      83ec50              sub esp, 0x50               ; 'P'
0x00408143      56                  push esi
0x00408144      57                  push edi
0x00408145      b90e000000          mov ecx, 0xe
0x0040814a      bed0134300          mov esi, str.http:__www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
0x0040814f      8d7c2408            lea edi, [esp + 8]
0x00408153      33c0                xor eax, eax
0x00408155      f3a5                rep movsd dword es:[edi], dword ptr [esi]
0x00408157      a4                  movsb byte es:[edi], byte ptr [esi]
0x00408158      89442441            mov dword [esp + 0x41], eax
0x0040815c      89442445            mov dword [esp + 0x45], eax
0x00408160      89442449            mov dword [esp + 0x49], eax
0x00408164      8944244d            mov dword [esp + 0x4d], eax
0x00408168      89442451            mov dword [esp + 0x51], eax
0x0040816c      6689442455          mov word [esp + 0x55], ax
0x00408171      50                  push eax
0x00408172      50                  push eax
0x00408173      50                  push eax
0x00408174      6a01                push 1                      ; 'Z.'
0x00408176      50                  push eax
0x00408177      88442466            mov byte [esp + 0x6b], al
0x0040817b      ff1534a14000        call dword [sym.imp.WININET.dll_InternetOpenA] ; 0x40a134
0x00408181      6a00                push 0
0x00408183      6800000084          push 0x84000000
0x00408188      6a00                push 0
0x0040818a      8d4c2414            lea ecx, [esp + 0x14]
0x0040818e      8bf0                mov esi, eax
0x00408190      6a00                push 0
0x00408192      51                  push ecx
0x00408193      56                  push esi
0x00408194      ff1538a14000        call dword [sym.imp.WININET.dll_InternetOpenUrlA] ; 0x40a138
0x0040819a      8bf8                mov edi, eax
0x0040819c      56                  push esi
0x0040819d      8b353ca14000        mov esi, dword sym.imp.WININET.dll_InternetCloseHandle ; [0x40a13c:4]=
0x004081a3      85ff                test edi, edi
0x004081a5      7515                jne 0x4081bc
0x004081a7      ffd6                call esi
0x004081a9      6a00                push 0
0x004081ab      ffd6                call esi
0x004081ad      e8defeffff          call sub.KERNEL32.dll_GetModuleFileNameA_90; dword GetModuleFileNameA(
0x004081b2      5f                  pop edi
0x004081b3      33c0                xor eax, eax
0x004081b5      5e                  pop esi
0x004081b6      83c450              add esp, 0x50               ; 'P'
0x004081b9      c21000              ret 0x10
    ; JMP XREF from 0x004081a5 (sub.WININET.dll_InternetOpenA_140)
0x004081bc      ffd6                call esi
0x004081be      57                  push edi
0x004081bf      ffd6                call esi
0x004081c1      5f                  pop edi
0x004081c2      33c0                xor eax, eax
0x004081c4      5e                  pop esi
0x004081c5      83c450              add esp, 0x50               ; 'P'
0x004081c8      c21000              ret 0x10
```

Demo


XORI
DASHBOARD
DISASM
EntryPoint
sub_404c1e
sub_402b1c
sub_405421
sub_401c1b
sub_4014aa
sub_401996
sub_401aae
sub_402f95
sub_4066b4
sub_40607a
sub_4071e4
sub_40487d
sub_4046e0
sub_403bc2
sub_406e7f
sub_4029e7
sub_4037fd
sub_406fcc
sub_405463
sub_4021fe
sub_40651a
sub_401a4f
sub_4029f1
sub_404319
sub_406f5e
sub_402a63
sub_406286
sub_4071f5
sub_4057ff
sub_4031a9
sub_401ea7
sub_4049df
sub_402436
EntryPoint
0x4052b7: push ebp
0x4052b8: jmp 0x401f59
0x401f59: mov ebp, esp
0x401f5b: jmp 0x405549
0x405549: and esp, 0xfffffff8
0x40554c: sub esp, 0xcc
0x405552: push ebx
0x405553: push esi
0x405554: push edi
0x405555: jmp 0x405f27
0x405f27: xor esi, esi
0x405f29: push 0xc4
0x405f2e: lea eax, [esp+0x18]
0x405f32: push esi
0x405f33: push eax
0x405f34: mov [esp+0x1c], esi
0x405f38: call 0x404c1e
0x405f3d: jmp 0x401d94
0x401d94: add esp, 0xc
0x401d97: lea edi, [esp+0x10]
0x401d9b: call 0x402b1c
0x401da0: test eax, eax
0x401da2: je 0x404d01
0x401da8: mov eax, edi
0x401daa: jmp 0x40560a
0x40560a: push eax
0x40560b: call 0x4014aa
0x405610: jmp 0x4019e0
0x4019e0: pop ecx
0x4019e1: cmp eax, 0x1
0x4019e4: je 0x405272
0x4019ea: mov eax, edi
0x4019ec: jmp 0x40526
Graph
Strings
Linear

github.com/endgameinc/xori

@malwareunicorn
@rseymour