ЯEVERSEU
A Web-based Reverse Engineering Platform

Steve Bumbera, Mathew Piscitelli, Ilya Shavrov, Adnan Sumra, Tyler Thomas

Advisors: Dr. Ibrahim Baggili; Liberty Page

Introduction

Reverse engineering is the practice of analyzing a system to understand the technical details of its inner workings. The origins of reverse engineering are rooted in hardware, but modern practices have extended into software. Software reverse engineering involves deconstructing an existing program in an attempt to understand how it works. Reverse engineering in the context of this project refers exclusively to software.

Reverse engineering is an extremely complex subject that is not only difficult to learn, but even more difficult to conduct in practice. This is due to the immense complexity of such a daunting task, as well as the lack of tools and resources tailored to new individuals attempting to enter the domain. This leads to a lack of qualified candidates who can respond to current security threats in both the public and private sectors.

Reverse engineering is a skill critical to national security from both an offensive and defensive perspective. In order to attack a system, one must understand how it works. Modern malware can be extremely complex, being able to compromise systems in ways previously never thought possible. The Department of Homeland Security's Cybersecurity and Infrastructure Security Agency (CISA) has labeled ransomware as a national crisis. Reverse engineering is critical for the detection and mitigation of these advanced threats.

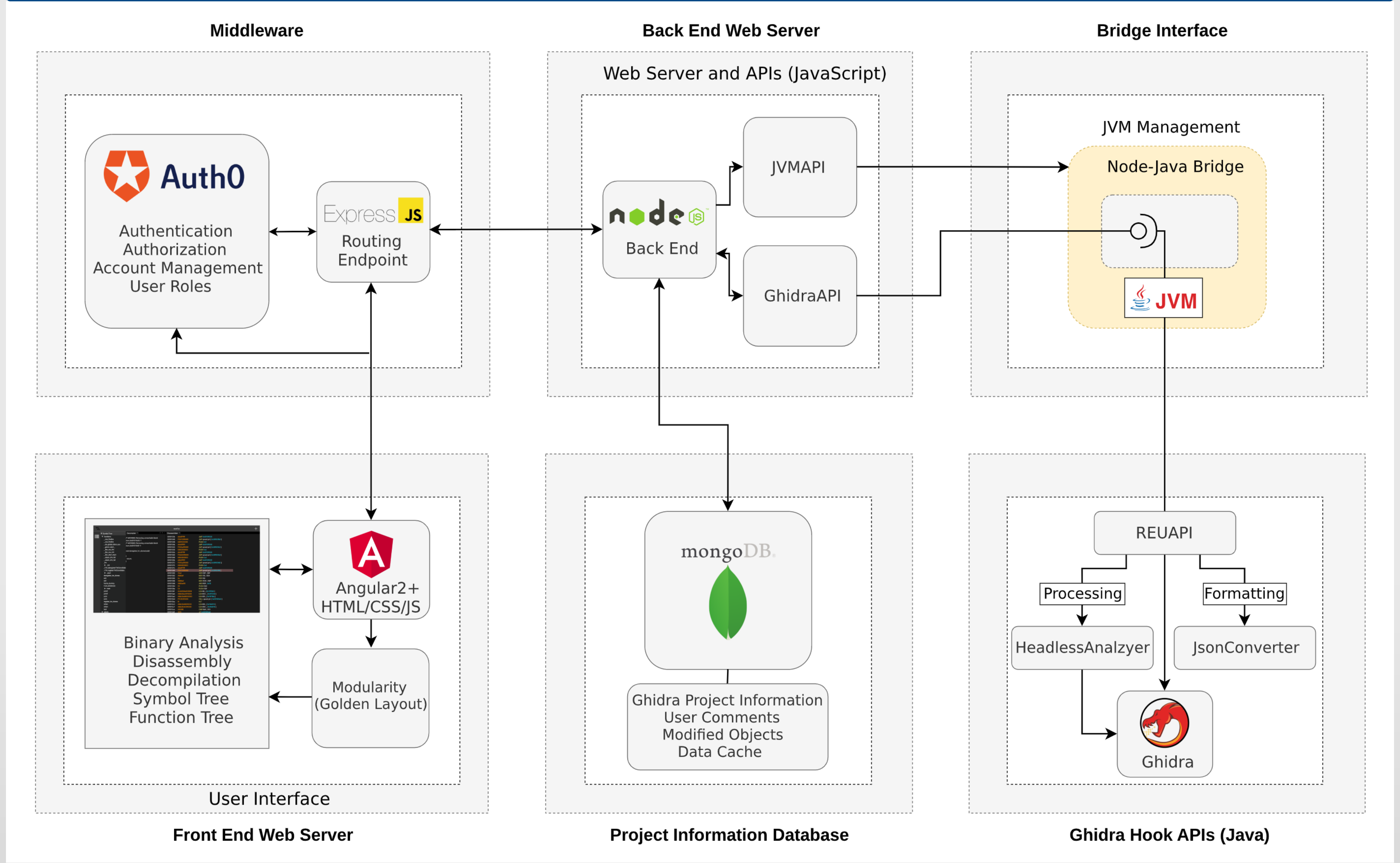
Technical Details

ReverseU's technology stack consists of a MEAN (MongoDB, Express, Angular and NodeJS) web server, a Java backend which hooks into the Ghidra application, and a bridge connecting the components. The software utilizes a set of application programming interfaces (APIs) to enable communication between technologies. Each instance of the Ghidra application is governed by a Java Virtual Machine (JVM). The execution environment is managed by our JavaScript JVMAPI using a Node-Java bridge library. Utilizing this bridge, our GhidraApi on the JavaScript side and REUAPI on the Java side pass information back and forth.

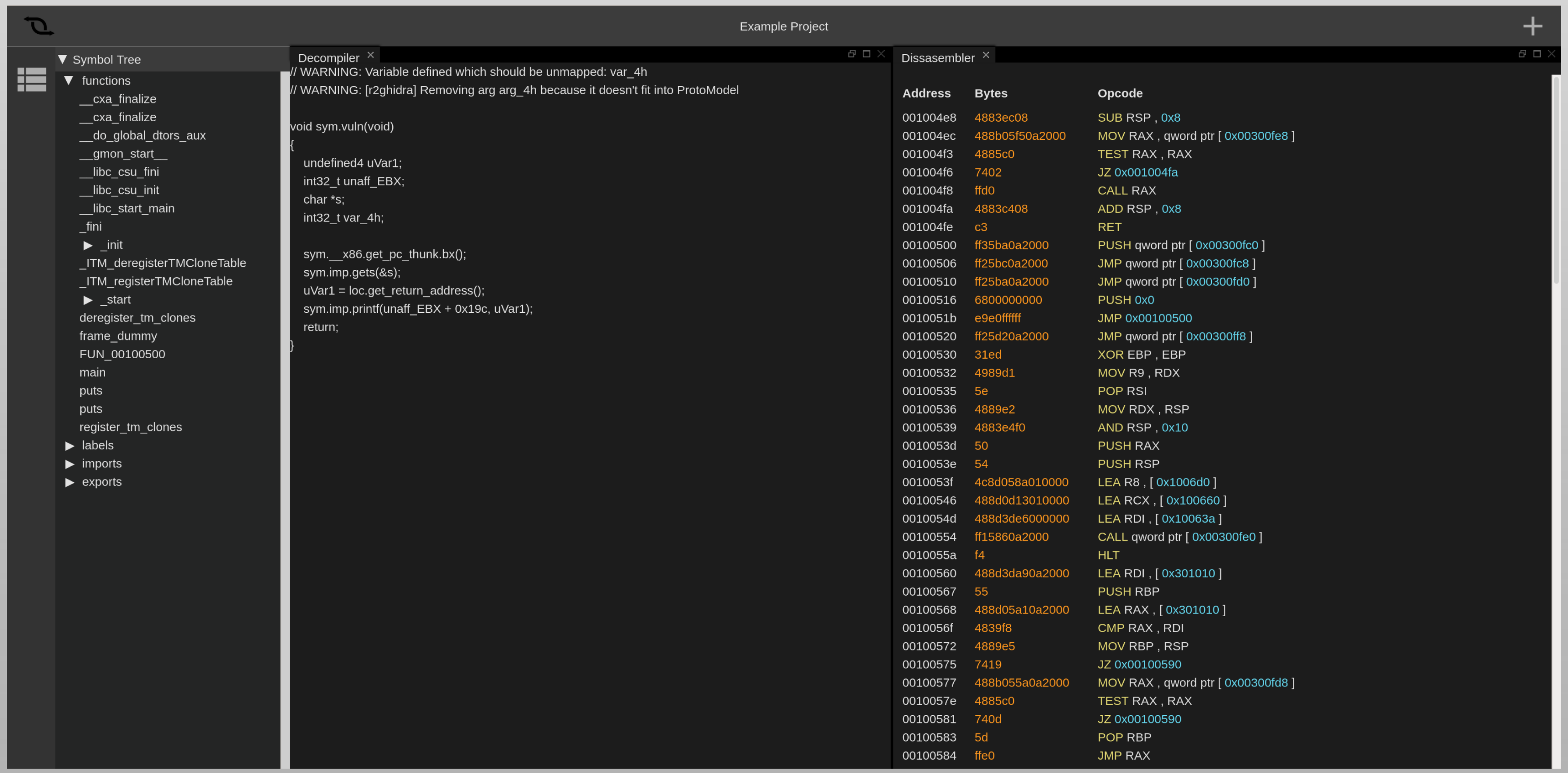
When a request for information is made by the user in the Angular interface, it passes through an ExpressJS server and is routed to an appropriate function on the back-end Node endpoint. Functions within the GhidraApi are called, which pass information to the REUAPI. Ghidra functionality is leveraged to perform analysis and return JSON output through the APIs to the Angular client. Using Golden Layout, a framework for creating resizable windows, the data is displayed intuitively to the user. Currently, ReverseU implements the disassembler, decompiler, defined strings, and symbol tree views present in Ghidra. In the future, ReverseU will implement and extend more advanced Ghidra features.

ReverseU leverages MongoDB and Auth0 for information storage. Ghidra-related user data such as project ownership is stored in a MongoDB database, managed by the Node server. MongoDB is also used as a cache to limit the number of API calls to the JVM. Sensitive information is handled by Auth0, a secure third-party authentication and authorization solution.

Project Design



Proof of Concept User Interface



```
▼ functions
    __cxa_finalize
    __cxa_finalize
    __do_global_dtors_aux
    __gmon_start__
    __libc_csu_fini
    __libc_csu_init
    __libc_start_main
    _fini
  ▶ _init
    _ITM_deregisterTMCloneTable
    _ITM_registerTMCloneTable
  ▶ _start
    deregister_tm_clones
    frame_dummy
    FUN_00100500
    main
    puts
    puts
    register_tm_clones
▶ labels
▶ imports
▶ exports
```

```
// WARNING: Variable defined which should be unmapped: var_4h
// WARNING: [r2ghidra] Removing arg arg_4h because it doesn't fit into ProtoModel

void sym.vuln(void)
{
  undefined4 uVar1;
  int32_t unaff_EBX;
  char *s;
  int32_t var_4h;

  sym.__x86.get_pc_thunk.bx();
  sym.imp.gets(&s);
  uVar1 = loc.get_return_address();
  sym.imp.printf(unaff_EBX + 0x19c, uVar1);
  return;
}
```

```
Address    Bytes            Opcode
001004e8   4883ec08         SUB RSP , 0x8
001004ec   488b05f50a2000   MOV RAX , qword ptr [ 0x00300fe8 ]
001004f3   4885c0           TEST RAX , RAX
001004f6   7402             JZ 0x001004fa
001004f8   ffd0             CALL RAX
001004fa   4883c408         ADD RSP , 0x8
001004fe   c3               RET
00100500   ff35ba0a2000     PUSH qword ptr [ 0x00300fc0 ]
00100506   ff25bc0a2000     JMP qword ptr [ 0x00300fc8 ]
00100510   ff25ba0a2000     JMP qword ptr [ 0x00300fd0 ]
00100516   6800000000       PUSH 0x0
0010051b   e9e0ffffff       JMP 0x00100500
00100520   ff25d20a2000     JMP qword ptr [ 0x00300ff8 ]
00100530   31ed             XOR EBP , EBP
00100532   4989d1           MOV R9 , RDX
00100535   5e               POP RSI
00100536   4889e2           MOV RDX , RSP
00100539   4883e4f0         AND RSP , 0x10
0010053d   50               PUSH RAX
0010053e   54               PUSH RSP
0010053f   4c8d058a010000   LEA R8 , [ 0x1006d0 ]
00100546   488d0d13010000   LEA RCX , [ 0x100660 ]
0010054d   488d3de6000000   LEA RDI , [ 0x10063a ]
00100554   ff15860a2000     CALL qword ptr [ 0x00300fe0 ]
0010055a   f4               HLT
00100560   488d3da90a2000   LEA RDI , [ 0x301010 ]
00100567   55               PUSH RBP
00100568   488d05a10a2000   LEA RAX , [ 0x301010 ]
0010056f   4839f8           CMP RAX , RDI
00100572   4889e5           MOV RBP , RSP
00100575   7419             JZ 0x00100590
00100577   488b055a0a2000   MOV RAX , qword ptr [ 0x00300fd8 ]
0010057e   4885c0           TEST RAX , RAX
00100581   740d             JZ 0x00100590
00100583   5d               POP RBP
00100584   ffe0             JMP RAX
```

Development

The first stage of development involved deciding on a data processor for reverse engineering tasks. Ghidra, written by the National Security Agency (NSA), was chosen as the sole disassembler and decompiler for ReverseU because it is an extremely powerful and open source tool. Comparable software such as IDA Pro can cost upwards of $7,000.

Using an open source data processor granted us the ability to access, modify, and extend current Ghidra functionality. We created an API by wrapping existing Ghidra classes in custom Java code. This design decision created many technical challenges. We gained total control over the inner workings of Ghidra, but only after delving through a dated Java code base containing over 500,000 lines and 9,000 classes. A significant portion of development time was spent investigating how to extract data from Ghidra and inject data into live Java objects from a web browser.

By using the Node-Java bridge, we can instantiate a JVM inside of a Node endpoint. This allows the Node backend to directly interface with Ghidra using thin API calls. Leveraging this REST API, we can extract data from Ghidra in JSON format and cache it in a database for quick access.

On the front end, the Angular application requests data from the REST API and has several responsibilities including making HTTP requests to the server, state management and providing a web interface to the user. The interface is designed to replicate Ghidra's functionility, while providing a more modern design.

Problems and Goals

Prior to development of ReverseU, it was necessary to explore existing reverse engineering tools and understand the pain points that users experience. We came to the conclusion that the user interface of current reverse engineering tools can can intimidating to beginners. Additionally, some tools are expensive, adding a barrier to entry for users.

The goals, features, and design choices of ReverseU were made to address these issues, as well as encourage new users to learn reverse engineering. The user interface was tailored to present data in a user-friendly way. Our modern yet simple GUI design contrasts with the dated look of other tools. ReverseU will be publicly available, allowing anyone to use the tool for free. Finally, the web-based nature of ReverseU improves accessibility to a reverse engineering environment by eliminating the need to install and maintain software.

Acknowledgements

The ReverseU Team would like to thank and acknowledge Professors Ibrahim Baggili and Liberty Page for their support and guidance during the development of ReverseU. Additionally, we would like to thank the National Security Agency (NSA) Research Directorate for creating and publicly releasing Ghidra as part of their Cybersecurity mission.