Enhancing Binary Analysis through Cognitive Load Theory

by

Sean Smits

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2022 by the
Graduate Supervisory Committee:

Ruoyu Wang, Co-Chair
Yan Shoshitaishvili, Co-Chair
Adam Doupé

ARIZONA STATE UNIVERSITY

May 2022

ABSTRACT

Reverse engineering is a process focused on gaining an understanding for the intricacies of a system. This practice is critical in cybersecurity as it promotes the findings and patching of vulnerabilities as well as the counteracting of malware. Disassemblers and decompilers have become essential when reverse engineering due to the readability of information they transcribe from binary files. However, these tools still tend to produce involved and complicated outputs that hinder the acquisition of knowledge during binary analysis. Cognitive Load Theory (CLT) explains that this hindrance is due to the human brain's inability to process superfluous amounts of data. CLT classifies this data into three cognitive load types — intrinsic, extraneous, and germane — that each can help gauge complex procedures.

In this research paper, a novel program call graph is presented accounting for these CLT principles. The goal of this graphical view is to reduce the cognitive load tied to the depiction of binary information and to enhance the overall binary analysis process. This feature was implemented within the binary analysis tool, `angr` and it's user interface counterpart, `angr-management`. Additionally, this paper will examine a conducted user study to quantitatively and qualitatively evaluate the effectiveness of the newly proposed proximity view (PV). The user study includes a binary challenge solving portion measured by defined metrics and a survey phase to receive direct participant feedback regarding the view. The results from this study show statistically significant evidence that PV aids in challenge solving and improves the overall understanding binaries. The results also signify that this improvement comes with the cost of time. The survey section of the user study further indicates that users find PV beneficial to the reverse engineering process, but additional information needs to be included in future developments.

DEDICATION

A little more than a year ago, I started my journey into the world of cybersecurity, clueless of its vastness and depth. Luckily, I had great navigational guides – Yan and Fish.

I distinctly remember the fear Yan struck into me in the first lecture of pwn.college. His cautionary words of the difficulty, required skills, and time commitment was unparalleled to any previous course I had taken. However, Yan's passion for the topics we planned to cover and his promise that his students can become sufficient "yellow belt" hackers inspired me to continue. After completing pwn.college, Yan invited me to join his more advanced course to start becoming a "real-world" hacker. This opportunity allowed me to harness the previous skills I learned and apply it directly to securing modern software services, ultimately leading me to acquiring my first CVE. To top it off, Yan invited me to join ASU's renowned SEFCOM Lab. Thank you Yan, for providing so many opportunities and accompanying me each step along the way.

The first time I met Fish I mistakenly took him for another student. I was asking for help on his class server and he reached out to me via a private message. Since this was the first experience I had in which a professor went out of his way to directly offer support, I assumed it was simply a sympathetic peer. Ever since that first embarrassing encounter, Fish has always been just a message away providing me with guidance and instruction, whether it be questions pertaining to his courses or merely a personal side project. Thank you Fish, for always making the time and giving me valuable advice I can continue to use throughout my career.

ACKNOWLEDGMENTS

I would like to further express my sincerest gratitude to my committee member, Professor Adam Doupé. Without Adam's guidance and personal efforts, this project would not have been possible. I would also like to thank Professor Tiffany Bao for providing valuable input and feedback throughout the course of this project. Additionally, I would like to thank the rest of the SEFCOM Lab for warmly welcoming me and providing me with continuous insight into our field. Lastly, I would like to thank my family and friends for their constant love and support throughout all my endeavors.

TABLE OF CONTENTS

Page

LIST OF TABLES vi

LIST OF FIGURES vii

CHAPTER

1 INTRODUCTION 1

2 BACKGROUND 3

2.1 Binary Analysis Tools 3

2.1.1 Disassembly & Decompilation 3

2.1.2 Control Flow Graph 4

2.1.3 IDA's Proximity Browser 5

2.2 Cognitive Load Theory 5

2.2.1 Information Filtering 6

2.2.2 Information Organization 6

3 PROBLEM STATEMENT 7

4 DESIGN 8

4.1 Overview 8

4.2 Proximity Call Graph 8

4.3 Proximity Nodes 9

4.4 User Interaction 11

5 IMPLEMENTATION 12

5.1 Preliminary Measures 12

5.2 Proximity Analysis 14

5.3 Proximity View 16

5.3.1 Blocks 16

5.3.2 Arrows 17

CHAPTER Page

5.3.3 Graph 17

5.3.4 View 18

6 USER STUDY 19

6.1 Participants 19

6.2 Experiment Environment 19

6.3 Challenges 20

6.4 Survey 21

7 EVALUATION 22

7.1 Results 22

7.1.1 Quantitative Results 22

7.1.2 Qualitative Results 26

7.2 Discussion 28

8 FUTURE WORK 29

9 CONCLUSION 31

REFERENCES 32

APPENDIX

A RECRUITMENT ANNOUNCEMENTS 34

B CHALLENGE QUESTIONS 36

C SURVEY QUESTIONS — PROXIMITY GROUP 38

D SURVEY QUESTIONS — CONTROL GROUP 40

E CONTROL GROUP RESPONSES 42

F IRB APPROVAL OF USER STUDY 44

LIST OF TABLES

Table Page

7.1 Control v. Proximity: Quantitative Metrics 24

7.2 Evaluating Statistical Significance 24

B.1 Challenge Questions 37

C.1 Proximity View Survey Questions 39

D.1 Control Survey Questions 41

LIST OF FIGURES

Figure Page

2.1 Disassembly View CFG 4

2.2 IDA's Proximity Browser 5

4.1 Proximity View CFG 10

4.2 Expanded *function_call* Nodes 11

5.1 AIL CFG 13

7.1 Participant Background Impact on Challenges 25

7.2 Participant Background Impact on Solve Times 26

7.3 Positive Feedback Questions 27

7.4 Negative Feedback Questions 27

D.1 Image Used in the Control Group Survey 41

E.1 Responses Regarding PV 43

E.2 Image Used in the Control Group Survey 43

Chapter 1

INTRODUCTION

Reverse engineering (RE) is a technique for obtaining knowledge about the inner workings of a system without the requirement of high-level instruction. This procedure is widely used in the field of cybersecurity and is viewed as a powerful defense to combat against cyberattacks. One reason for RE's effectiveness is due to the difference in analysis perspective. Rather than focusing on software through traditional program syntax review, understanding is acquired through the examination of low-level internal building blocks. With this narrow and magnified scope, many of the details that would otherwise be invisible become more apparent. This is particularly relevant when dealing with binaries. Binaries are compiled, executable code files that are interpretable by computers and other programs, but not readily identifiable externally. Because of this, RE is an essential process when investigating binaries.

Binary analysis tools have been developed to partially automate the RE process. These tools are capable of translating illegible binary byte sequences to a comprehensible medium through disassembly and decompilation. Despite this, the complexity of the generated outputs and the plethora of information provided can induce cognitive overload for users [14]. An oversupply of information, even when containing meaningful details, can deluge the working memory [14]. Consequently, this inhibits reverse engineers' ability to process data and properly acquire knowledge - the fundamentals of the RE process.

In this paper, a newly designed control flow graph is proposed to address this cognitive overload problem. The intention behind this graph view, titled proximity view (PV), is to allow users to quickly acquire information and insight about a decompiled binary through the application of techniques based on Cognitive Load Theory. Furthermore, the goal of the presented program call graph is to ultimately improve the binary analysis process.

This paper also includes the examination of said view through a 41 participant user study. These participants, ranging from students with a minimal background in RE to active professionals with years of experience, were tasked with solving 3 Capture the Flag (CTF) style RE challenges. Questions pertaining to the participants' understanding of the challenges together with their overall perceptions of PV were gathered throughout the study. The results from this study provide statistically significant evidence that PV improved participants' understanding scores and the amount of solved challenges. However, the results also indicate that participants that solved challenges using PV were on average slower than those without the view. The overall feedback signified users found PV useful and easy to use, but lacking in some relevant information.

Chapter 2

BACKGROUND

This section briefly covers necessary concepts for the understanding of proximity view and its application in the reverse engineering process of binary analysis.

2.1 Binary Analysis Tools

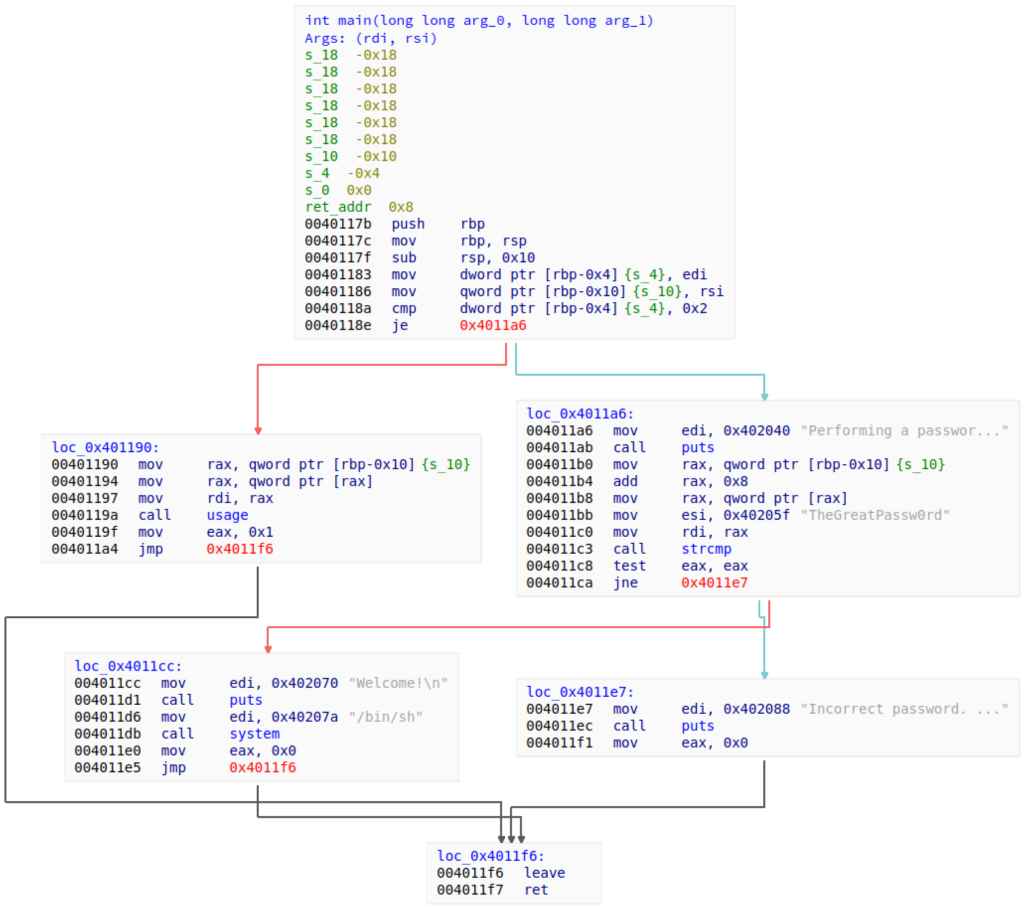
Binary analysis tools, often called disassemblers or decompilers, are developed software programs that aid in the process of reverse engineering compiled files by extracting and translating binary data into a high-level language. Hex-Rays Decompiler (`IDA Pro`) [11], `angr` [18], `rev.ng` [8], and `Bap` [2] are just a few of the tools actively used within the field of binary analysis research. Further research of these tools has lead to the development of `Phoenix` [3], `Dream` [27], and `Dream++` [26]. Each of these works focus on effectively converting binary information into a simplified and readable medium while preserving accurate control-flow of the program. This is done through the processes of disassembly and decompilation.

2.1.1 Disassembly & Decompilation

Disassembly is a procedure of recovering a symbolic representation of a program's assembly instructions from its binary form by mapping hexadecimal values back to instruction mnemonics [16]. Decompilation is the complex process of reconstructing a program to a high-level language from a low-level language [10]. Put simply, decompilation is the reverse engineered method of compilation.

2.1.2 Control Flow Graph

One commonly used data structure to represent program control flow is a control flow graph (CFG) [5, 6]. A CFG, or call graph, is simply a directed graph (digraph) composed of various types of nodes and directed edges. These digraphs are utilized in binary analysis as it signifies all traversal paths between code blocks and outlines the hierarchy of a program. These graphs are multi purposed data structures that 1) readily allow for optimizations to improve decompiler output [5, 6, 11, 18, 12, 26, 27, 3] and 2) provides users with an organized visual representation of information.



```
int main(long long arg_0, long long arg_1)
Args: (rdi, rsi)
s_18  -0x18
s_18  -0x18
s_18  -0x18
s_18  -0x18
s_18  -0x18
s_18  -0x18
s_10  -0x10
s_4  -0x4
s_0  0x0
ret_addr  0x8
0040117b  push    rbp
0040117c  mov     rbp, rsp
0040117f  sub     rsp, 0x10
00401183  mov     dword ptr [rbp-0x4] {s_4}, edi
00401186  mov     qword ptr [rbp-0x10] {s_10}, rsi
0040118a  cmp     dword ptr [rbp-0x4] {s_4}, 0x2
0040118e  je      0x4011a6
```

```
loc_0x401190:
00401190  mov     rax, qword ptr [rbp-0x10] {s_10}
00401194  mov     rax, qword ptr [rax]
00401197  mov     rdi, rax
0040119a  call    usage
0040119f  mov     eax, 0x1
004011a4  jmp     0x4011f6
```

```
loc_0x4011a6:
004011a6  mov     edi, 0x402040 "Performing a passwor..."
004011ab  call    puts
004011b0  mov     rax, qword ptr [rbp-0x10] {s_10}
004011b4  add     rax, 0x8
004011b8  mov     rax, qword ptr [rax]
004011bb  mov     esi, 0x40205f "TheGreatPassw0rd"
004011c0  mov     rdi, rax
004011c3  call    strcmp
004011c8  test    eax, eax
004011ca  jne     0x4011e7
```

```
loc_0x4011cc:
004011cc  mov     edi, 0x402070 "Welcome!\n"
004011d1  call    puts
004011d6  mov     edi, 0x40207a "/bin/sh"
004011db  call    system
004011e0  mov     eax, 0x0
004011e5  jmp     0x4011f6
```

```
loc_0x4011e7:
004011e7  mov     edi, 0x402088 "Incorrect password. ..."
004011ec  call    puts
004011f1  mov     eax, 0x0
```

```
loc_0x4011f6:
004011f6  leave
004011f7  ret
```

Figure 2.1: Disassembly View CFG

2.1.3 IDA's Proximity Browser

`IDA Pro`'s proximity browser introduced a unique call graph focused on visualizing the relationships between functions, variables, and constants. Unlike the traditional disassembly CFG where nodes house assembly instructions, the browser's decompiled nodes only portray callee and caller functions, global variables, and constants as shown below in Figure 2.2 [15].

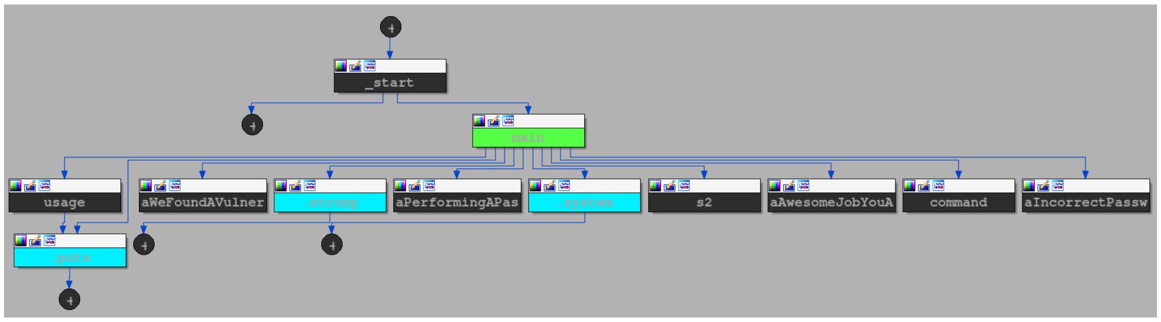


Figure 2.2: IDA's Proximity Browser

2.2 Cognitive Load Theory

In order to perceive the relevance of the proposed proximity view, it is essential gain a general comprehension of Cognitive Load Theory.

All cognitive processes, the mental action of gaining knowledge and comprehension, rely on working memory. Working memory is recognized as a system in which information gathered in the short-term is temporarily held and processed before being stored into long-term memory. This same system is responsible for focusing and engaging in problem solving tasks [19]. Cognitive load theory (CLT) states there is a limited amount of information that can be effectively handled due to the restricted capacity of an individual's working memory [20, 21, 22, 23, 24, 25]. Hence, cognitive load is a measurement of the amount of information that can held in working memory. Furthermore, CLT explains that cognitive overload leads to the inability to properly process, understand, and retain information [20, 21, 22, 23, 24, 25]. During cognitive processing, cognitive load is categorized into three types: (a) intrinsic load, the representation of the difficulty and complexity of a task (b) extraneous load, the presentation of irrelevant information and (c) germane load, the construction of a knowledge schema (a conceptual system for understanding) for the long-term memory [7, 20, 21, 22, 23, 24, 25]. The reduction of cognitive load is achieved through several methods, however those relevant to the study will solely be covered.

2.2.1 Information Filtering

One of the most straightforward techniques for managing cognitive overload is information filtering (IF). Since cognitive load increases with respect to the amount of information presented, filtering out extraneous data (i.e. reducing extraneous load) will inherently decrease the demand on the working memory [20, 21, 22, 23, 24, 25]. IF is also important as it prioritizes obtaining quality, applicable information over the availability of more information [17].

2.2.2 Information Organization

Another method for limiting cognitive load is the organization of complex material. Information that is presented in a manner that registers with a user's schema can be readily incorporated into their knowledge base [17]. On the contrary, unfamiliar formatting of portrayed information can add to cognitive load and cause confusion. Put simply, data categorized in a manner that a user is accustomed to promotes a greater understanding of the material.

Chapter 3

PROBLEM STATEMENT

Reverse engineering binaries is a process that inherently has a high intrinsic load due to the complexity entailed in analyzing code from a low to high level. Numerous studies [27, 26, 5, 6, 4, 9, 10, 3, 11, 18, 12] have been conducted that explore improving this process through optimizing disassembly and decompilation. However, binary analysis tools still contribute to high extraneous loads when portraying many lines of intricate assembly instructions or obscure decompiled code. This extensive demand on the working memory leads to cognitive overload and inhibits reverse engineers' ability to examine, extract, and evaluate information. The focal point of this paper is on reducing the high intrinsic and extraneous cognitive loads rampant within binary analysis to enhance this reverse engineering process.

Chapter 4

DESIGN

In this section, a novel optimized proximity view is proposed which provides reverse engineers a simplified and interactable graphical representation of a binary. The goal of this feature is to enhance the binary RE process through the CLT methods of information filtering and information organization. More specifically, the view's design prioritizes providing quality relevant data, removing extraneous information, and presenting output in a format conducive to common RE knowledge schemata.

4.1 Overview

The proposed `angr-management` proximity view harnesses a similar concept of `Ida Pro`'s proximity browser with respect to CLT, leading to several deviations in call graph design. It should be noted that PV is a feature intended to be paired with other views within the `angr-management` framework as it is inherently designed to limit the amount of presented information to accelerate general understanding of a binary. From this gathered knowledge, reversers can then formulate assumptions about the binary and begin to look at code blocks under a more detailed view.

4.2 Proximity Call Graph

Under the lens of CLT, CFGs are an effective organizational architecture to use for PV [13]. By creating a knowledge schema familiar to reverse engineers (i.e. a CFG), germane load will be increased promoting a greater understanding. Furthermore, this systematic structuring makes it easier to comprehend code which, in turn, reduces intrinsic load.

Kruegel *et al.* [16], Andriesse *et al.* [1], to name a few, each highlight the importance of CFGs when recovering the symbolic representation of a program's assembly instructions from its binary form. Due to the significance of these generated CFGs in the disassembly process, it is counter intuitive to disregard this structural integrity when designing a simplified view such as PV. Rather, the graph integrity should be upheld and extraneous information should be properly filtered. This is the first distinct design difference when compared to `Ida Pro's` proximity browser. Since edges are the skeleton of the CFG, they should remain intact. The nodes, however, contain an excess of information that can be refined.

4.3 Proximity Nodes

Functions are essential pieces to understanding code functionality, hence it is important to include them in PV. Intuitively, functions are represented by *function* and *function_call* nodes within the CFG. *Function* nodes, or parent nodes, are the caller functions within the CFG. *Function_call* nodes, or children nodes, are the callee functions that are decedents from the ancestral *function* node in the CFG.

Variables and constants are also imperative to comprehending code, however, these are trickier to include in the PV schema. As PV is primarily focused on reducing intrinsic and extraneous cognitive load, the inclusion of all instructions utilizing variables and constants would be counter productive. Instead, PV filters this information and recognizes variables and constants as relevant only when as arguments for *function_call* nodes. The inclusion of this information only in the *function_call* nodes also differs from `Ida Pro's` proximity browser. This new design provides users with a high-level insight into what variables and constants maybe used for without overloading the working memory. Moreover, the intrinsic load is reduced as removing variable and constant nodes from the control flow of the program is more intuitive to follow.

As general instruction semantics are regularly ignored when creating *function* and *function_call* nodes, the same should remain true for code segments without function calls. However, these blocks cannot be entirely disregarded since the structure of the CFG should be maintained, as addressed above. This justifies the addition of an *empty* node as shown below in Figure 4.1.

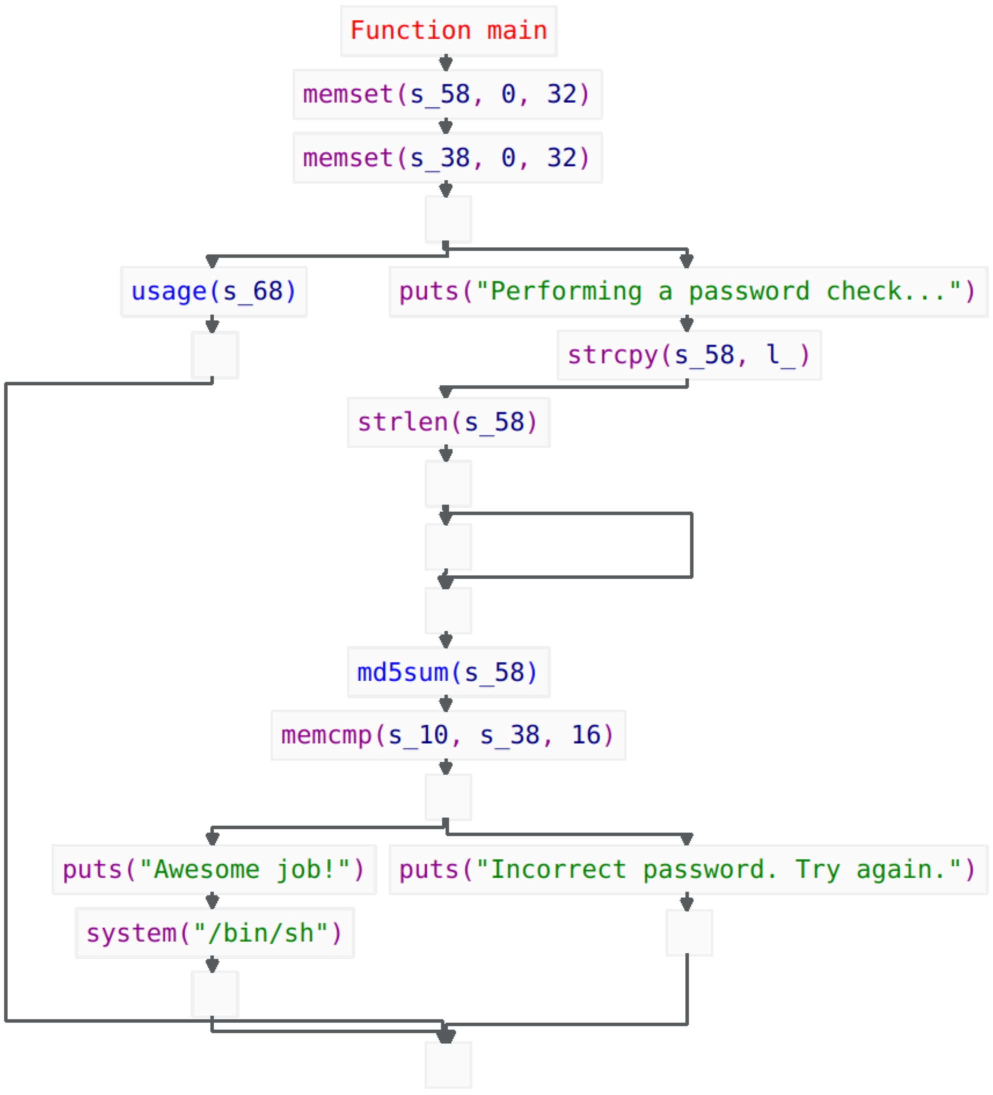
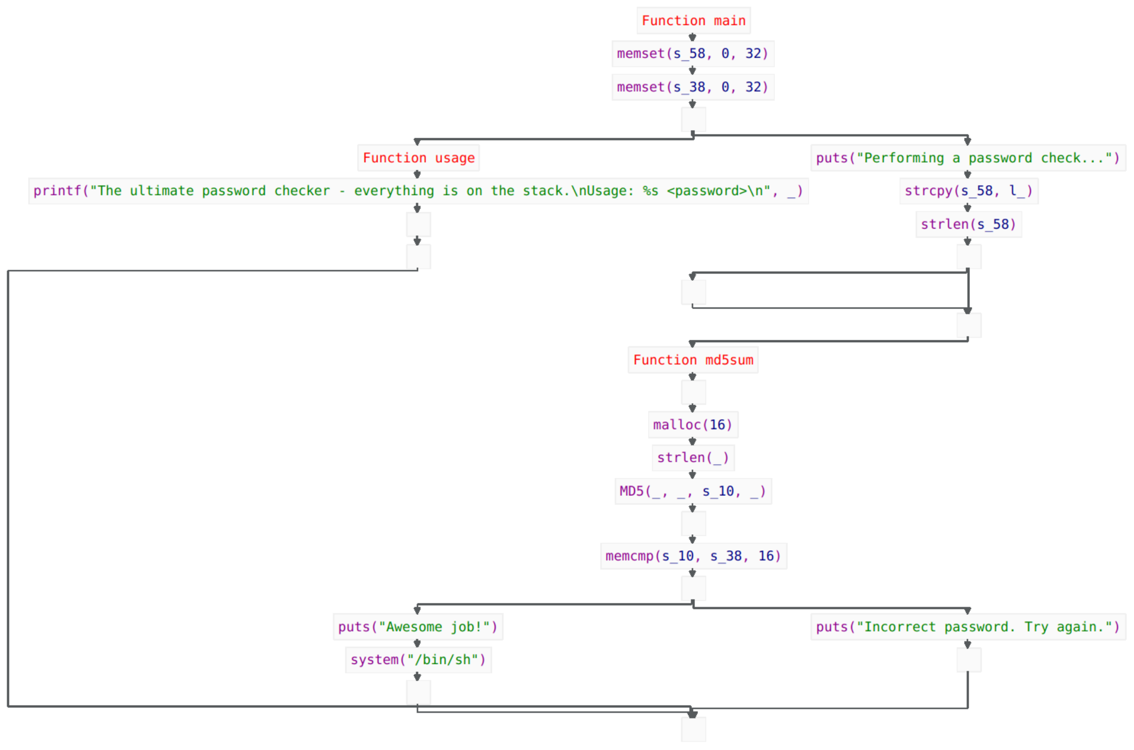


Figure 4.1: Proximity View CFG

4.4 User Interaction

The proposed user interaction of PV is also designed to aid in the RE process. By creating an interactable CFG, users will have the ability to move the graph within the window and adjust magnification when needed. This interactable control graph also expedites navigation. Users can select CFG nodes within PV they wish to examine and PV will redirect them to the correlating code block in the disassembly view. This directly supports the notion of pairing multiple `angr-management` views to gain knowledge through different perspectives. Additionally, when hovering over PV nodes, all incoming and outgoing edges will be highlighted to show all traversal paths. PV also supports *function_call* node expansion and collapsing. This allows reverse engineers to run proximity analysis on multiple functions within the same view, and to close them when desired. A visual of this is provided in Figure 4.2 below.



```
Function main
memset(s_58, 0, 32)
memset(s_38, 0, 32)
Function usage
printf("The ultimate password checker - everything is on the stack.\nUsage: %s <password>\n", _)
puts("Performing a password check...")
strcpy(s_58, l_)
strlen(s_58)
Function md5sum
malloc(16)
strlen(_)
MD5(_, _, s_10, _)
memcmp(s_10, s_38, 16)
puts("Awesome job!")
system("/bin/sh")
puts("Incorrect password. Try again.")
```

Figure 4.2: Expanded *function_call* Nodes

Chapter 5

IMPLEMENTATION

Within this section, the initial implementation of the proposed design framework of proximity view is discussed. The implementation of PV can be broken down into two primary components, the analysis that generates the proximity graph within `angr`, and the graphical user interface (GUI) view that presents the analysis within `angr-management`. Both components total to approximately 500 lines of contributed Python 3 code. These elements will be addressed in the following Proximity Analysis and Proximity View sections respectively. General functionality for preexisting `angr` analyses and `angr-management` features utilized in development will be explained for contextual purposes.

5.1 Preliminary Measures

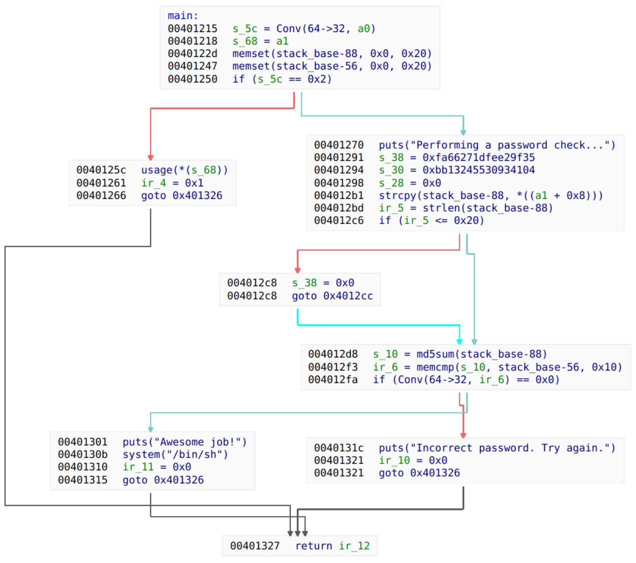
Before any proximity analysis can be conducted, first a binary must be translated into a interpretable representation. Within the `angr` library, disassembly is conducted through the `CFG` analysis. This analysis will disassemble a given binary into assembly language blocks represented in the form of a CFG.

```
>>> import angr
# load a binary
>>> proj = angr.Project("hello_world", auto_load_libs=False)
<Project hello_world>
# disassemble binary
>>> cfg = proj.analyses.CFG()
<CFG Analysis Result at 0x7fc2483390a0>
```

From here, further simplification happens through `angr`'s `Decompiler` analysis. This analysis takes in a given function and the disassembly, and will output a decompiled, high-level language.

```
# load the main function
>>> func = cfg.kb.functions['main']
<Function main (0x401160)>
# decompile
>>> dec = proj.analyses.Decompiler(func, cfg=cfg.model)
<ProximityGraphAnalysis Analysis Result at 0x7fc2385869a0>
```

During the process of decompilation, it is important to note that `angr` first creates an intermediate representation called *angr intermediate language* (AIL). As the name suggests, AIL is a form in between low and high level language. Because of this, AIL also has a CFG (shown in Figure 5.1) directly corresponding to the disassembly CFG.



```
main:
00401215  s_5c = Conv(64->32, a0)
00401218  s_68 = a1
0040122d  memset(stack_base-88, 0x0, 0x20)
00401247  memset(stack_base-56, 0x0, 0x20)
00401250  if (s_5c == 0x2)
```

```
0040125c  usage(*(s_68))
00401261  ir_4 = 0x1
00401266  goto 0x401326
```

```
00401270  puts("Performing a password check...")
00401291  s_38 = 0xfa66271dfee29f35
00401294  s_30 = 0xbb13245530934104
00401298  s_28 = 0x0
004012b1  strcpy(stack_base-88, *((a1 + 0x8)))
004012bd  ir_5 = strlen(stack_base-88)
004012c6  if (ir_5 <= 0x20)
```

```
004012c8  s_38 = 0x0
004012c8  goto 0x4012cc
```

```
004012d8  s_10 = md5sum(stack_base-88)
004012f3  ir_6 = memcmp(s_10, stack_base-56, 0x10)
004012fa  if (Conv(64->32, ir_6) == 0x0)
```

```
00401301  puts("Awesome job!")
0040130b  system("/bin/sh")
00401310  ir_11 = 0x0
00401315  goto 0x401326
```

```
0040131c  puts("Incorrect password. Try again.")
00401321  ir_10 = 0x0
00401321  goto 0x401326
```

```
00401327  return ir_12
```

Figure 5.1: AIL CFG

Finally, the target function, the disassembled CFG, and the decompilation can be passed to the proximity analysis.

```
>>> prox_graph = proj.analyses.Proximity(func, cfg.model, cfg.kb.xrefs,
↪  decompilation=dec)
<ProximityGraphAnalysis Analysis Result at 0x7fc2385869a0>
```

5.2 Proximity Analysis

Proximity analysis will first create its own digraph utilizing the `networkx` library. The target function, in this case ‘main’, will be the root *function* node as described in the design. Throughout the rest of the analysis function, information from the provided data structures will be scanned and passed to nodes within the PV graph.

Since `angr` essentially parses the disassembly to translate to AIL, it makes sense to utilize the AIL CFG generated to gather relevant data. This eliminates the need to create a new function to parse the disassembly or the decompiled high-level language. In order to process this data, each AIL instruction within every AIL node will be checked and any relevant information found will be temporarily stored. Traversing the AIL instructions is done through the `angr` class `AILBlockWalker`.

```
bw = AILBlockWalker()
for ail_edge in ail_graph.edges:
    for ail_node in ail_edge:
        bw.walk(ail_node)
        ...
```

Within the `AILBlockWalker`, all types of AIL statements and expressions are handled. By PV design, call statements and call expressions are the only handlers that need modification. When a call statement or call expression is met by the `AILBlockWalker`, the handler must check all of the arguments for variables, strings, and constants. After this check, a new *function_call* node will be created which will include any arguments if found. Once the block walker finishes handling all statements and expressions, the found *function_call* nodes and their arguments can be added to the PV graph in their calling order. In the event that the `AILBlockWalker` fails to find any call statements or expressions within an AIL node, an *empty* node must be added.

```python
    ...
    if self.handled_stmts:
        # Add each handled stmt to the graph in calling order
        for idx, current in enumerate(self.handled_stmts):
            if idx > 0:
                graph.add_edge(self.handled_stmts[idx - 1], current)
        if block == ail_edge[0]:
            proxi_node = self.handled_stmts[-1]
        else:
            proxi_node = self.handled_stmts[0]
        self.handled_stmts = []
    else:
        proxi_node = BaseProxiNode(ProxiNodeTypes.Empty, {block.addr})
    new_edge += (proxi_node,)
graph.add_edge(*new_edge)
```

Now that the core logic of proximity analysis is covered, the action of function call expansion can be understood. When a user triggers the expansion event in the front end, proximity analysis is rerun with an additional list argument containing the target functions to expand. When those functions are met by the `AILBlockWalker`, a *function* node is created instead of a *function_call* node. After the initial PV graph is generated, a new PV sub-graph will be created for the *function* node. Once both graphs are created, a connector function is called to insert the sub-graph into the main PV graph. First, the nodes succeeding the *function* node are temporarily stored. Next, the edges from the *function* node to its successors are removed and a new edge is drawn from the *function* node to the root node of the sub-graph. Finally, all sub-graph end nodes are then pointed to the previously stored list of successor nodes.

5.3 Proximity View

Proximity View is the interactive medium that presents all of the information gathered from the proximity analysis. As the content within the CFG has been altered in the backend, much of the `angr-management` code for the preexisting digraphs within views can be reused. PV can be better conceptualized by understanding each of its working parts from low to high level. This includes the the nodes within the graph, the arrows between nodes, the visual of the graph inside the view, and the view itself within the `angr-management` window.

5.3.1 Blocks

As previously covered, PV supports 3 primary nodes: *function*, *function_call*, and *empty*. *Empty* nodes are represented by the base graph block, an empty block with set dimensions. This block class also has several event listeners. One listener checks for mouse double clicks; when this event occurs the disassembly view will be opened to node with an address corresponding to the clicked block. Other listeners check for when a mouse is hovering over a block; this information gets relayed back to PV to be handled.

Function nodes are represented by function blocks which inherit the characteristics and events from the base block. These function blocks simply contain the text of a caller function name. All function blocks in a graph have been passed through proximity analysis. Due to this, function blocks have a listener that checks for `CTRL + Double-Click`. When this event is triggered, proximity view collapses the proximity subgraph of that function block and returns it to a normal call block.

Lastly, the *function_call* nodes are represented by call blocks. These blocks also inherit from the base block. Within the call block, the callee function name is shown as well as all arguments. These arguments include strings, integers, stack variables, and unresolvable arguments. Call blocks also have an added event listener that checks for `CTRL + Double-Click`. When this event is triggered, the listener expands the the call function and generates a proximity view for the triggered function block.

5.3.2 Arrows

To signify control flow between blocks, graph arrows are created to represent edges. These graph arrows have already been implemented for all other CFGs and can be further applied to PV. These arrows essentially have length and bent corners based on their direction and distance between nodes.

5.3.3 Graph

The interactable graph is an essential piece to PV as it presents all the valuable information gathered from proximity analysis. The proximity graph class inherits from the `QZoomableDraggbleGraphicsView` which is an `angr-management` base graph class that supports graph movement and magnification. The proximity graph class then passes all graph blocks to `angr-management`'s `GraphLayouter`. This layouter organizes the graph based on node sizes and passed margin values. Finally, arrows are generated between each laid out block node.

5.3.4 View

The view itself is the last component that ties all the pieces together. The proximity view class inherits from the `BaseView`. The `BaseView` is a base class which defines the characteristics of a main window view within `angr-management`. When the proximity view is requested, initialization of a graph widget occurs. This widget is essentially an empty initialized proximity graph. PV then decompiles the focused function and passes this to the proximity analysis in the backend. A CFG representation of the target function is then output. PV then iterates over all the nodes in the CFG and converts them to graph blocks. The proximity graph, which is also the graph widget, then orients the blocks and draws directed edge arrows. The graph widget is then reloaded and a visual representation of the target function's CFG is visible.

One unique event that gets handled in the view is user hovering. When the mouse enters within a block's dimensions, the hover enter event will passed from block to view to the graph widget (i.e. proximity graph). Proximity graph then checks finds all in-edge arrows and out-edge arrows of that block to be highlighted. Similarly, when the mouse leaves the block's dimensions, the hover leave event will continually passed until the graph widget removes the highlighted effect on the arrows.

Chapter 6

USER STUDY

In this section, the organization of the user study will be discussed. The goal of this study was to test the impact of proximity view on the binary analysis process. In order to both quantitatively and qualitative assess the feature's effectiveness, participants were tasked with solving 3 challenge problems alongside taking a feedback survey to measure user perception and satisfaction.

6.1 Participants

As this feature focuses on improving the complex process of binary analysis, a certain level of knowledge is required to properly evaluate proximity view and to solve the challenges. For this reason, specific language was used when conducting outreach as shown in Appendix A. Participants with valid experience that expressed interest were then sent a unique session key to access the web-based experiment platform.

6.2 Experiment Environment

Within the experiment platform, participants were provided with a sequence of pages including: a consent form, a study overview, Remote Desktop Protocol (RDP) connection instructions, an introduction on `angr-management`, 3 challenge question pages, and a feedback survey. Upon session key approval, participants were randomly assigned to either a control group or a proximity group, and the order of their challenge questions were shuffled. These actions were taken to eliminate any bias in group assignment and task order.

When users were ready to connect to a Virtual Machine (VM), the website would clone a controlled image instance and provide credentials to the participant. Inside the VM, participants were restricted to guest privileges, preventing them from installing other tools or damaging the experiment environment. Furthermore, an `angr-management` wrapper was developed for the experiment to control the sequence of presented challenges, and to limit participants to specific views. Participants within the control group had restricted access only to `angr-management`'s disassembly, functions, strings, and hex views. Members of the proximity group were permitted the same views with the addition of the newly developed proximity view. All other aspects of the VM and `angr-management` were permitted, unless specifically stated in the instructions. Participants' VMs were recorded to ensure adherence to guidelines and the legitimacy of given answers.

Upon successful connection, the website then provided users with further instruction of `angr-management`. Participants in the proximity group received additional guidance regarding the concept of proximity view and its features. When ready, participants could then proceed to the challenge solving phase.

6.3 Challenges

The participants' task was to analyze each presented binary and determine a specific input that would cause the file to output an answer flag. All answers abided by the format `flag{...}` to be clearly identifiable by participants. Participants were allotted 10 minutes to solve each challenge, however, if desired, they were permitted to continue reversing once time expired. Each participant, regardless of group, was presented with the same 3 challenges in a randomized sequence. These challenges include: a) quad — an RE challenge that forces users to find an integer password, mathematically solve (using the quadratic equation) for two additional integers, and find a string password b) letters — a binary that checks for a specific range of bytes that must be even valued and totals the bytes ordinal value compared to `0xbeef` and c) maze — a convoluted function maze in which users can select which function to go to in attempt to find the “win” function.

After each challenge, participants were asked to provide details explaining their working process and the input needed if they were successful in reversing the binary. These questions can be found within Table B.1 on page 37.

6.4 Survey

Following the completion of the last challenge, participants were then given a feedback survey to measure user perception and satisfaction. Both groups were asked the same background survey questions pertaining to user expertise. Control group members were then shown an example proximity view analysis and asked questions regarding its perceived relevance (see Table D.1 on page 41). Proximity group members were instead asked questions regarding the usability of the view and its effectiveness (see Table C.1 on page 39).

Chapter 7

EVALUATION

In this section, the study results pertaining to the conducted user study are presented and discussed.

7.1 Results

A total of 78 session key invitations were sent out to individuals that expressed interest in the study. Of these distributed keys, the target goal was getting 30 participants.

51 individuals used their unique session key and began the study. 10 of these participants stopped midway through the experiment and their data has been removed, leaving a total of 41 participants that completed the experiment in its entirety.

Of the 41 participants, 18 (44%) were randomly assigned the control group and 23 (56%) were assigned to proximity group. Within the control group, 11 participants identified as having 1 year or less experience in RE compared to the 7 that claimed to have 2 or more years experience. In the proximity group, 13 stated to have 1 year or less RE experience and 10 identified as having 2 years or more. For information readability, it should be noted those who claimed to have 1 year or less experience are labelled as *Beginners* and those who claimed to have 2 or more years of RE experience are labelled as *Experts*.

7.1.1 Quantitative Results

To better understand the impact of PV on binary analysis, results are categorized into the following metrics: *Overall Understanding*, *Overall Completion*, and *Comple-*

tion Solve Time.

Overall Understanding is measured through a calculated score. Each of the participant's responses were evaluated and assigned a value of 0, 0.5, or 1. Participants that were unable to solve the challenge and did not show a significant understanding of the challenge were given a 0. Those participants that could not find the correct input, but provided an in-partial answer and signified a majority comprehension were awarded a 0.5. Lastly, those that were able to solve the challenge with an explanation were scored a 1. The scoring of these challenges was conducted without the knowledge of group type to eliminate any bias.

Overall Completion is the metric used to signify participants ability to completely solve each of the challenges by providing a correct input to output the flag.

Completion Solve Time metrics are the recorded times for these completely solved challenges. It is important to note the scoring metrics and solve time metrics are kept as separate results since partial solves awarded with 0.5 are not proper indicators of solve time.

To test the statistical significance of PV's positive influence on these metrics, the null hypothesis for each metric states the means between proximity group and control group are equal. The alternative hypotheses for *Overall Understanding* and *Overall Completion* state the mean of proximity group will be greater than control group. The alternative hypothesis for *Completion Solve Times* states the mean of proximity group will be less than the control group. For 95% confidence, the probability error level of each metric will be set to $\alpha = 0.05$.

CONTROL — 18 Participants				
Challenges	*Avg. Score*	*Complete Solves*	*Complete Solve %*	*Avg. Solve Time*
Quad	75.00%	13	72.22%	576.96 s
Letters	11.11%	0	0.00%	-
Maze	38.89%	5	27.78%	593.63 s
PROXIMITY — 23 Participants				
Challenges	*Avg. Score*	*Complete Solves*	*Complete Solve %*	*Avg. Solve Time*
Quad	89.13%	20	86.96%	639.80 s
Letters	17.39%	0	0.00%	-
Maze	67.39%	14	60.87%	983.75 s

Table 7.1: Control v. Proximity: Quantitative Metrics

Table 7.1 above summarizes the results of the experiment. Participants that were permitted the use of proximity view successfully solved more challenges (higher *Overall Completion*) and received a higher score for each challenge (greater *Overall Understanding*). Interestingly, this came with the cost of time spent to solve the challenge (higher *Completion Solve Time*).

	Overall Understanding	**Overall Completion**	**Completion Solve Time**
Test	2 Sample T-Test	2 Proportion Z-Test	2 Sample T-Test
Alpha	$\alpha = 0.05$	$\alpha = 0.05$	$\alpha = 0.05$
Null Hypothesis	$H_0 : \mu_p = \mu_c$	$H_0 : \mu_p = \mu_c$	$H_0 : \mu_p = \mu_c$
Alternative Hypothesis	$H_1 : \mu_p > \mu_c$	$H_1 : \mu_p > \mu_c$	$H_1 : \mu_p < \mu_c$
p-value	0.0246	0.0378	0.9291
Outcome	Reject Null Hypothesis	Reject Null Hypothesis	Accept Null Hypothesis

Table 7.2: Evaluating Statistical Significance

Table 7.2 (see above), breaks down the statistical significance of the set quantitative metrics. Since the null hypothesis was rejected for both *Overall Understanding* and *Overall Completion*, there is sufficient evidence to conclude that the PV improved the average scoring and increased the mean completion of challenges. However, since the null hypothesis was accepted for *Completion Solve Time*, there is insufficient evidence to claim PV participants have a lower average solve time.

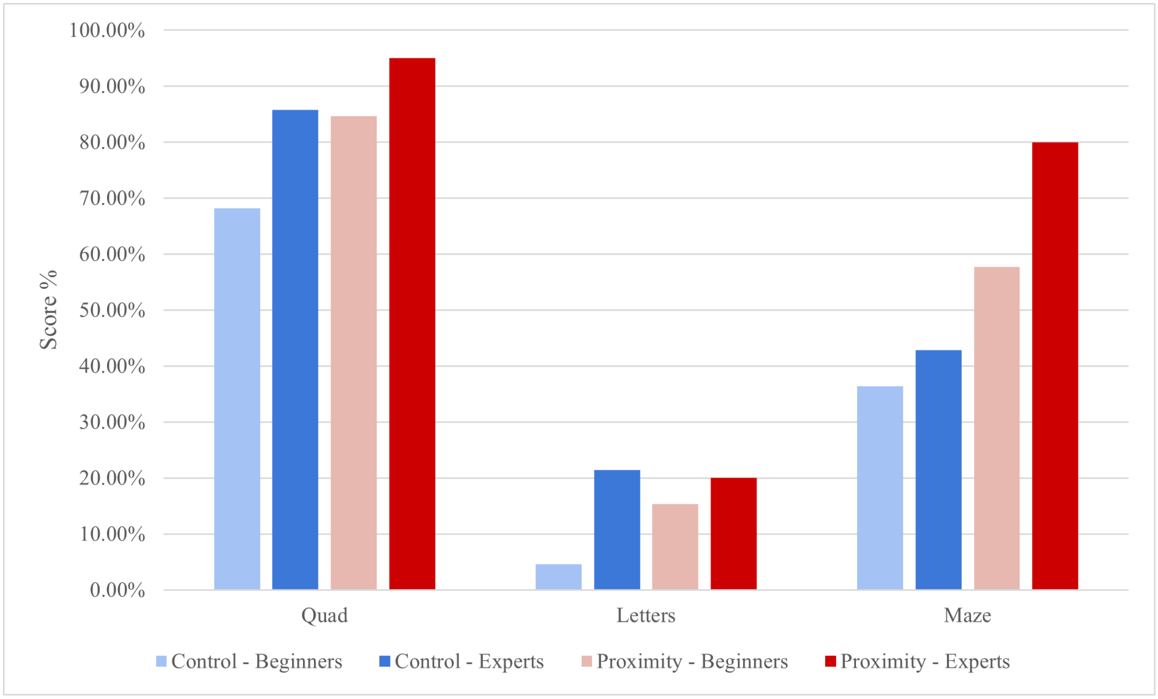


Figure 7.1: Participant Background Impact on Challenges

Figure 7.1 (see above) highlights another observation pertaining to RE experience. Participants with proximity view that had 1 year or less experience (*Beginners*), nearly matched control group *Experts* (2 or more years of RE experience) in Quad score and out performed them in solving Maze. Similarly, in Figure 7.2 (see below), the solve times are shown with respect to these experience groups. The letters challenge was redacted since there were no complete solves. The Maze times were particularly interesting as the control *Experts* had significant lower solve times, and proximity group *Beginners* had considerably longer solve times. This graph explains how control group participants, on average, solved Quad 1 minute and Maze 6.5 minutes faster than those with PV.

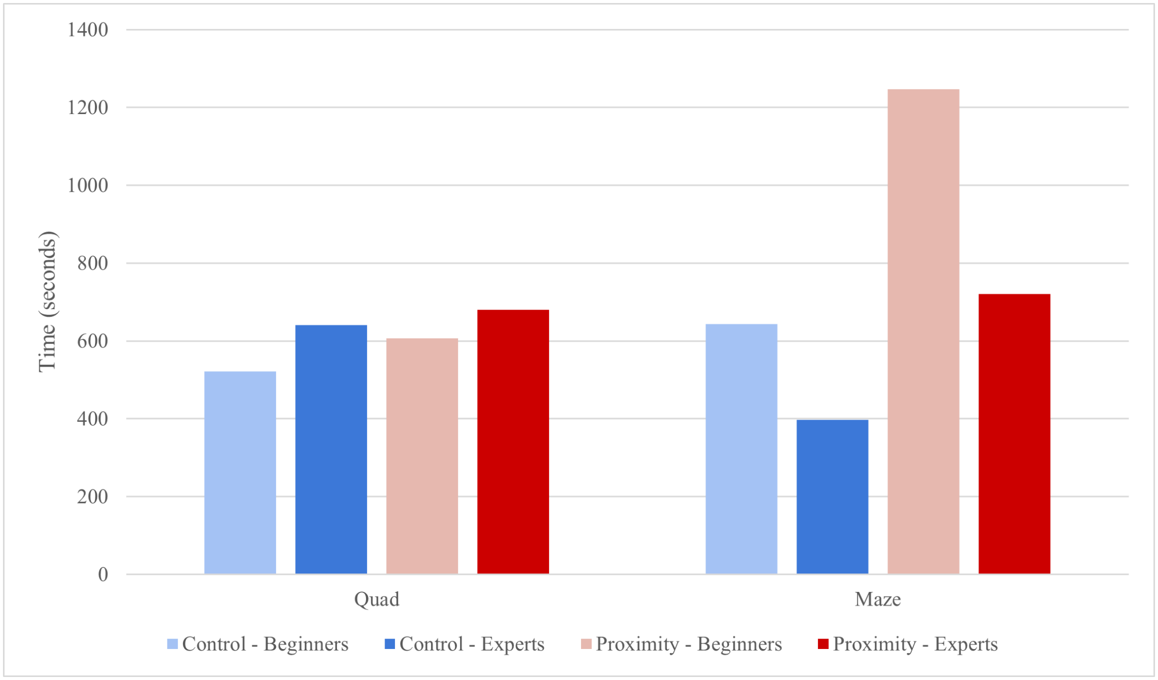


Figure 7.2: Participant Background Impact on Solve Times

7.1.2 Qualitative Results

In order to get insight beyond raw data, participants were also asked questions pertaining to their perceptions of proximity view. In Figure 7.3 and Figure 7.4 (see below), positive sentiment towards PV can be seen in the dark and light green sections whereas negative sentiment can be seen in the light and dark read areas. Overall, feedback for PV was positive on most fronts including usability, utility, and readability. Despite this, user's widely agreed that PV is lacking valuable information. Control group participants also expressed positive interest in PV as shown in Figure E.1 in Appendix E.

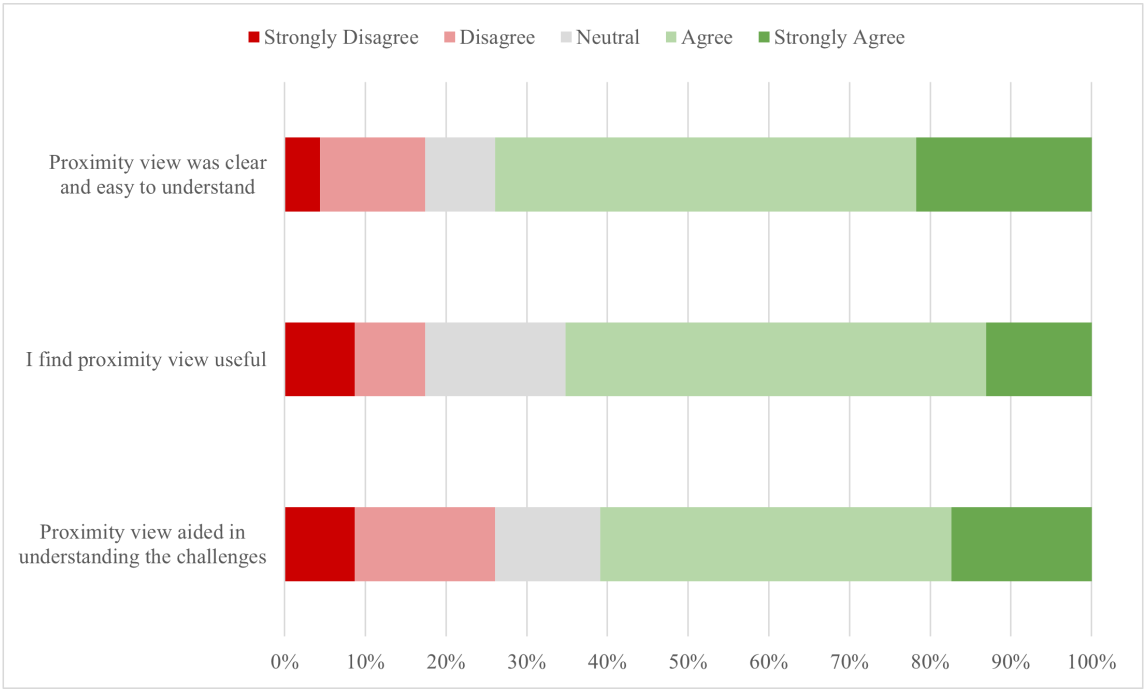


Figure 7.3: Positive Feedback Questions

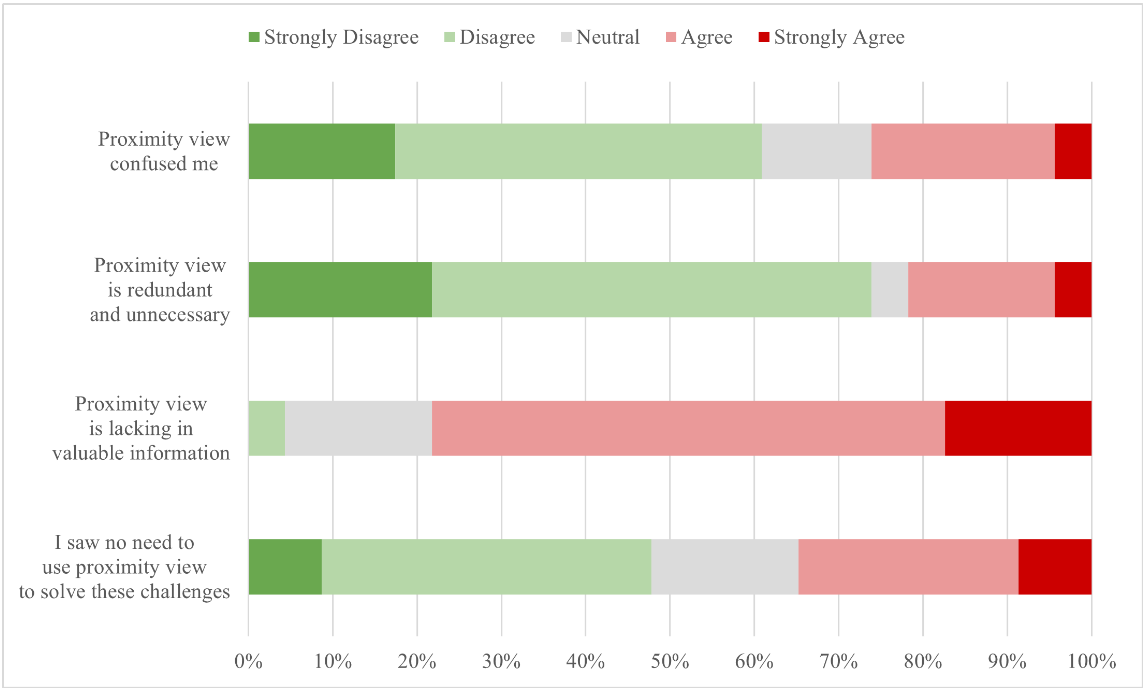


Figure 7.4: Negative Feedback Questions

7.2 Discussion

Although the 95% confidence level accounts for 5% of error, there are several experiment factors to mention when accounting for this resulting data. The first factor to mention is the Letters challenge. This challenge was intended to be one step greater in difficulty compared to Quad, however, with the suggested 10 minute time limit, no participants were able to solve. A pilot study was taken to test the entire framework along with the challenges, but this was an unpredictable outcome that could not have been caught by the test run.

Another factor to consider lies within the Maze challenge. The Maze challenge was designed to be a complicated web of functions in which users could not easily follow. Due to a design flaw in the challenge, there was an unintended solution that introduced an unintentional bias. Many participants provided with PV used the view to solve the challenge by viewing control flow and finding the correct traversal path. However, control group participants lacking this view had to revert to alternative methods of solving. Because of this, many control participants, especially the *Experts*, found the unintended solution which expedited the solving process, ultimately lowering Completion Solve Times.

The last factor relates back to the web-based experiment platform. Since the website and the VMs were being hosted on a single server in North America, there were several occasions in which the server faced too much demand and crashed. A few participants were actively taking the study during these occasional crashes, hence their data has been removed due to incompleteness. Similarly, some participants noted difficulties when solving challenges due to latency when connected to the server hosted VMs which also could potentially impact the results.

Chapter 8

FUTURE WORK

As the presented proximity view in this paper explores improving binary analysis through methods described in CLT, there are numerous avenues to continue research in this regard.

One area in particular that can be investigated is the presentation of information specific to PV. As the idea behind PV is to quickly provide an overview and a general understanding of a given binary without overwhelming a user, the current design filters information that is deemed irrelevant in this context. Empty nodes, for example, were implemented to hide blocks of code lacking any function calls, as these details would likely be explored later in the process of reverse engineering. Feedback from several participants in the user study suggested that including *some* details within these empty blocks, such as cmp instructions for branch conditions, may aid in understanding control flow. Future research should be conducted to determine 1) the amount of information that can be presented without inducing cognitive overload and 2) whether the inclusion of these details within PV proves beneficial.

Another research avenue that should be explored is the user interactability of proximity view and the impact this has on productivity. Several participants from the user study blindly agreed that adding the ability to toggle disassembled code within PV's CFG nodes would be more beneficial than being redirected to a disassembly view. Multiple participants also suggested implementing an action that finds traversal paths between specific nodes and generates a new CFG. Future work can be done to create these features and examine their effects.

Further studies can also be conducted to examine the influence of binary dimensions and complexity on the effectiveness of proximity view. In this study, 3 different binaries were tested to see if PV proved useful in each instance. The intricacy and size of these binaries was limited as each challenge was intended to be solved within a 10 minute time frame. However, this view may prove to be more useful for high complexity binaries with a larger code base as cognitive overload is more likely to occur. More research can be done to determine the legitimacy of this claim.

Lastly, proximity view is only one proposed solution to the immense cognitive demand of binary analysis. Other binary analysis tools, such as disassembly or decompilation, should be studied through the lens of Cognitive Load Theory.

Chapter 9

CONCLUSION

This paper first discussed the complex process of reverse engineering binaries and the immense cognitive load inherently associated with it. Second, through the principles of information filtration and information organization derived from Cognitive Load Theory, a newly designed program call graph was proposed. The primary objective of this implemented tool was to enhance the process of understanding functionality through reducing complexity and cognitive load for users. This paper continued to examine the efficacy of a new proximity view on binary analysis through a conducted user study. The user study provided statistically significant data that highlighted the improvement in challenge understanding and the increase in challenges solved due to PV. Contrarily, the results also suggested participant solve times were prolonged because of the use of the view. Overall, participants regarded PV as a useful and easy to use tool that aided in binary analysis, but needs slight modification to the amount of presented information.

REFERENCES

[1] Andriesse, D., X. Chen, V. Van Der Veen, A. Slowinska and H. Bos, "An in-depth analysis of disassembly on full-scale x86/x64 binaries", in "25th USENIX Security Symposium (USENIX Security 16)", pp. 583–600 (2016).

[2] Brumley, D., I. Jager, T. Avgerinos and E. J. Schwartz, "Bap: A binary analysis platform", in "International Conference on Computer Aided Verification", pp. 463–469 (Springer, 2011).

[3] Brumley, D., J. Lee, E. J. Schwartz and M. Woo, "Native x86 decompilation using semantics-preserving structural analysis and iterative control-flow structuring", in "22nd USENIX Security Symposium (USENIX Security 13)", pp. 353–368 (2013).

[4] Chang, B.-Y. E., M. Harren and G. C. Necula, "Analysis of low-level code using cooperating decompilers", in "International Static Analysis Symposium", pp. 318–335 (Springer, 2006).

[5] Cifuentes, C., *Reverse compilation techniques* (Citeseer, 1994).

[6] Cifuentes, C., D. Simon and A. Fraboulet, "Assembly to high-level language translation", in "Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)", pp. 228–237 (IEEE, 1998).

[7] DeLeeuw, K. E. and R. E. Mayer, "A comparison of three measures of cognitive load: Evidence for separable measures of intrinsic, extraneous, and germane load.", Journal of educational psychology **100**, 1, 223 (2008).

[8] Di Federico, A., M. Payer and G. Agosta, "rev. ng: a unified binary analysis framework to recover cfgs and function boundaries", in "Proceedings of the 26th International Conference on Compiler Construction", pp. 131–141 (2017).

[9] Engel, F., R. Leupers, G. Ascheid, M. Ferger and M. Beemster, "Enhanced structural analysis for c code reconstruction from ir code", in "Proceedings of the 14th International Workshop on Software and Compilers for Embedded Systems", pp. 21–27 (2011).

[10] Fokin, A., E. Derevenetc, A. Chernov and K. Troshina, "Smartdec: approaching c++ decompilation", in "2011 18th Working Conference on Reverse Engineering", pp. 347–356 (IEEE, 2011).

[11] Guilfanov, I., "Decompilers and beyond", Black Hat USA **9**, 46 (2008).

[12] Gussoni, A., A. Di Federico, P. Fezzardi and G. Agosta, "A comb for decompiled c code", in "Proceedings of the 15th ACM Asia Conference on Computer and Communications Security", p. 637–651 (ACM, 2020), URL `https://dl.acm.org/doi/10.1145/3320269.3384766`.

[13] Huang, W., P. Eades and S.-H. Hong, “Measuring effectiveness of graph visualizations: A cognitive load perspective”, Information Visualization **8**, 3, 139–152 (2009).

[14] Kirsh, D., “A few thoughts on cognitive overload”, Intellectica **1**, 30 (2000).

[15] Koret, J., “New feature in ida 6.2: The proximity browser”, URL `https://hex-rays.com/blog/new-feature-in-ida-6-2-the-proximity-browser/` (2011).

[16] Kruegel, C., W. Robertson, F. Valeur and G. Vigna, “Static disassembly of obfuscated binaries”, in “USENIX security Symposium”, vol. 13, pp. 18–18 (2004).

[17] Quiroga, L. M., M. E. Crosby and M. K. Iding, “Reducing cognitive load”, in “37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the”, pp. 9–pp (IEEE, 2004).

[18] Shoshitaishvili, Y., R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel and G. Vigna, “SoK: (State of) The Art of War: Offensive Techniques in Binary Analysis”, in “IEEE Symposium on Security and Privacy”, (2016).

[19] Sorden, S. D., “A cognitive approach to instructional design for multimedia learning.”, Informing Science **8** (2005).

[20] Sweller, J., “Cognitive load during problem solving: Effects on learning”, Cognitive science **12**, 2, 257–285 (1988).

[21] Sweller, J., “Evolution of human cognitive architecture”, Psychology of learning and motivation **43**, 216–266 (2003).

[22] Sweller, J., “The redundancy principle in multimedia learning”, The Cambridge handbook of multimedia learning pp. 159–167 (2005).

[23] Sweller, J., “Cognitive bases of human creativity”, Educational Psychology Review **21**, 1, 11–19 (2009).

[24] Sweller, J., “Element interactivity and intrinsic, extraneous, and germane cognitive load”, Educational psychology review **22**, 2, 123–138 (2010).

[25] Sweller, J., “Cognitive load theory”, in “Psychology of learning and motivation”, vol. 55, pp. 37–76 (Elsevier, 2011).

[26] Yakdan, K., S. Dechand, E. Gerhards-Padilla and M. Smith, “Helping johnny to analyze malware: A usability-optimized decompiler and malware analysis user study”, in “2016 IEEE Symposium on Security and Privacy (SP)”, pp. 158–177 (2016).

[27] Yakdan, K., S. Eschweiler, E. Gerhards-Padilla and M. Smith, “No more gotos: Decompilation using pattern-independent control-flow structuring and semantic-preserving transformations.”, in “NDSS”, (Citeseer, 2015).

APPENDIX A

RECRUITMENT ANNOUNCEMENTS

When recruiting student participants in Arizona State University affiliated Discord servers, the following message [1] was posted:

Have you taken ASU's CSE 365, CSE 466, or CSE 545 or do you have a background in reverse engineering, debugging, and exploiting binaries? If you answered "yes" to either of these questions, you are invited to participate in a research study conducted by Arizona State University! You will be asked to solve various challenges using a tool called angr management in order to research the effectiveness of new features on the platform. Prior experience with angr management is not necessary. The research study is approximately 1.75 hours (100 minutes) and can be taken at your time of choice. Your participation will be rewarded with a $50 Amazon gift card upon completion. Participation in this study is voluntary.

When recruiting participants in other reverse engineering related Discord servers, the following message[1] was posted:

Do you have a background in reverse engineering, debugging, and exploiting binaries? If you answered "yes", you are invited to participate in a research study conducted by Arizona State University! You will be asked to solve various challenges using a tool called angr management in order to research the effectiveness of new features on the platform. Prior experience with angr management is not necessary. The research study is approximately 1.75 hours (100 minutes) and can be taken at your time of choice. Your participation will be rewarded with a $50 Amazon gift card upon completion. Participation in this study is voluntary.

[1]Contact information has been redacted from the message

APPENDIX B

CHALLENGE QUESTIONS

Table B.1: Challenge Questions

Questions	**Answers**		
Have you seen this challenge before?	Yes	No	I prefer not to answer
Briefly describe what you did during this challenge (bullet point explanation is acceptable):	*[text box]*		
Were you able to solve this challenge?	Yes	No	I prefer not to answer
What input(s) did you provide to solve this challenge?	*[text box]*		

APPENDIX C

SURVEY QUESTIONS — PROXIMITY GROUP

Table C.1: Proximity View Survey Questions

Questions	**Answers**					
How many years experience do you have in reverse engineering?	None	Less than 1 year	1 year	2 years	2+ years	I prefer not to answer
What is your perceived reverse engineering skill level?	Novice	Beginner	Competent	Proficient	Expert	I prefer not to answer
Have you used angr management before your participation today?	Yes		No		I prefer not to answer	
What is your perceived comfort level with angr management?	Novice	Beginner	Competent	Proficient	Expert	I prefer not to answer
I am sure that I correctly understood what the code of each challenge does	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
I saw no need to use proximity view to solve these challenges	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
Proximity view aided in understanding the challenges	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
Proximity view is lacking in valuable information	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
I find proximity view useful	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
Proximity view is redundant and unnecessary	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
Proximity view was clear and easy to understand	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
Proximity view confused me	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer

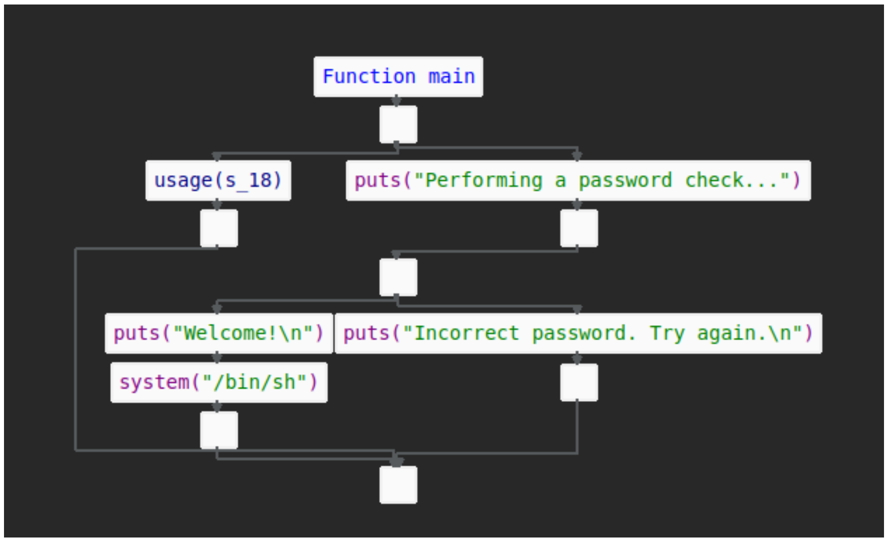
Feedback - If any, what improvements could be made to make proximity view more user friendly?

APPENDIX D

SURVEY QUESTIONS — CONTROL GROUP

Table D.1: Control Survey Questions

Questions	Answers					
How many years experience do you have in reverse engineering?	None	Less than 1 year	1 year	2 years	2+ years	I prefer not to answer
What is your perceived reverse engineering skill level?	Novice	Beginner	Competent	Proficient	Expert	I prefer not to answer
Have you used angr management before your participation today?	Yes		No		I prefer not to answer	
What is your perceived comfort level with angr management?	Novice	Beginner	Competent	Proficient	Expert	I prefer not to answer
I am sure that I correctly understood what the code of each challenge does	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
I found it difficult to understand these challenges	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
An overview of the challenges like in the image below would have been helpful	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer
This view in the image below seems confusing and unnecessary	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	I prefer not to answer



```
usage(s_18)
puts("Performing a password check...")
puts("Welcome!\n")
puts("Incorrect password. Try again.\n")
system("/bin/sh")
```

Figure D.1: Image Used in the Control Group Survey

APPENDIX E

CONTROL GROUP RESPONSES

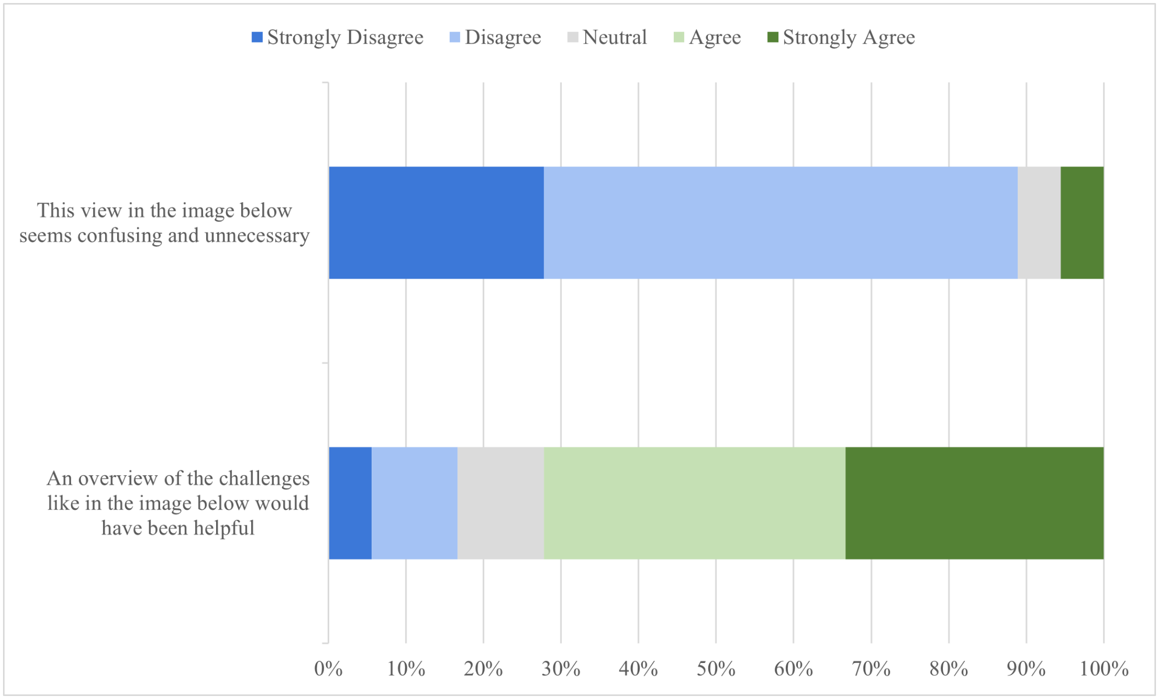
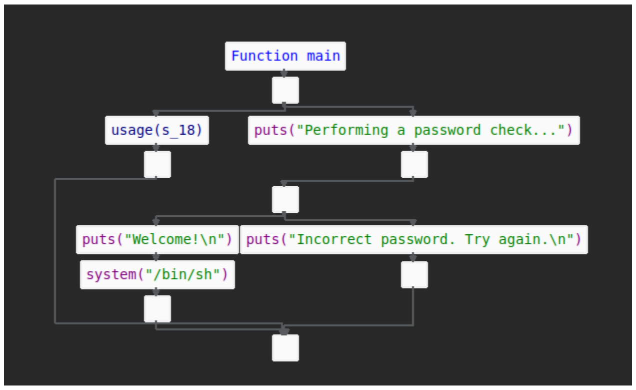


Figure E.1: Responses Regarding PV



Function main
usage(s_18)
puts("Performing a password check...")
puts("Welcome!\n")
puts("Incorrect password. Try again.\n")
system("/bin/sh")

Figure E.2: Image Used in the Control Group Survey

APPENDIX F

IRB APPROVAL OF USER STUDY

EXEMPTION GRANTED

Adam Doupe
SCAI: Computing and Augmented Intelligence, School of
-
doupe@asu.edu

Dear Adam Doupe:

On 2/21/2022 the ASU IRB reviewed the following protocol:

Type of Review:	Initial Study
Title:	Expediting Binary Analysis Through Data Dependency Graphs and Proximity Control Flow Graphs
Investigator:	Adam Doupe
IRB ID:	STUDY00015332
Funding:	Name: DOD: Defense Advanced Research Projects Agency (DARPA), Funding Source ID: FP00017167
Grant Title:	*CHECRS: Cognitive Human Enhancements for Cyber Reasoning Systems*
Grant ID:	*FP00017167*
Documents Reviewed:	• Consent_Form, Category: Consent Form; • DARPA Proposal, Category: Sponsor Attachment; • Debugging and Vulnerability Challenge Questions, Category: Measures (Survey questions/Interview questions /interview guides/focus group questions); • Debugging_and_Vulnerability_Survey, Category: Measures (Survey questions/Interview questions /interview guides/focus group questions); • Entire Experiment Text Outline, Category: Participant materials (specific directions for them); • Instructions / Greeting, Category: Recruitment Materials; • IRB Form, Category: IRB Protocol; • Recruitment Message, Category: Recruitment Materials; • Reverse Engineering Challenge Questions,

	Category: Measures (Survey questions/Interview questions /interview guides/focus group questions); • Reverse_Engineering_Survey, Category: Measures (Survey questions/Interview questions /interview guides/focus group questions);

The IRB determined that the protocol is considered exempt pursuant to Federal Regulations 45CFR46 (2) Tests, surveys, interviews, or observation, (3)(i)(A) - benign behavioral interventions on 2/21/2022. As a part of IRB review, scientific merit was considered.

In conducting this protocol you are required to follow the requirements listed in the INVESTIGATOR MANUAL (HRP-103).

If any changes are made to the study, the IRB must be notified at research.integrity@asu.edu to determine if additional reviews/approvals are required. Changes may include but not limited to revisions to data collection, survey and/or interview questions, and vulnerable populations, etc.

REMINDER - Effective January 12, 2022, in-person interactions with human subjects require adherence to all current policies for ASU faculty, staff, students and visitors. Up-to-date information regarding ASU's COVID-19 Management Strategy can be found here. IRB approval is related to the research activity involving human subjects, all other protocols related to COVID-19 management including face coverings, health checks, facility access, etc. are governed by current ASU policy.

Sincerely,

IRB Administrator

cc: Sean Smits
Sean Smits
Zeming Yu
Adam Doupe
Ruoyu Wang
Bailey Capuano