INVESTIGATING COLLABORATION IN SOFTWARE REVERSE ENGINEERING

THESIS

Allison M. Wong, 2d Lt, USAF

AFIT-ENG-MS-22-M-075

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-22-M-075

INVESTIGATING COLLABORATION IN SOFTWARE REVERSE ENGINEERING

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Allison M. Wong, B.S.

2d Lt, USAF

March 24, 2022

AFIT-ENG-MS-22-M-075

INVESTIGATING COLLABORATION IN SOFTWARE REVERSE ENGINEERING

THESIS

Allison M. Wong, B.S.
2d Lt, USAF

Committee Membership:

Lt Col Wayne C. Henry, Ph.D
Chair

Dr. Scott R. Graham, Ph.D
Member

Dr. Michael E. Miller, Ph.D
Member

AFIT-ENG-MS-22-M-075

Abstract

Reverse engineering (RE) is a rigorous process of exploration and analysis to support software design recovery and exploit development. The process is often conducted in teams to divide the workload and take full advantage of engineers' individual expertise and strengths. Collaboration in RE requires versatile and reliable tools that can match the environment's unpredictable and fluid nature. While studies on collaborative software development have indicated common best practices and implementations, similar standards have not been explored in reverse engineering.

This research conducts semi-structured interviews with reverse engineering experts to understand their needs and solutions while working in a team. The results describe an array of major challenges that are each addressed by employing tools such as issue tracking software, shared workspaces, and version control systems. Such tools support documentation and continuity, while mitigating redundancies in concurrent work. Though the value of these tools is acknowledged by the experts, seamless workflow integration remains a challenge. The identification of current needs and practices offers additional opportunities for collaborative tool developers to aid reverse engineers.

Table of Contents

Page

Page

List of Figures

List of Tables

INVESTIGATING COLLABORATION IN SOFTWARE REVERSE ENGINEERING

I. Introduction

1.1 Background and Problem Space

In a 2021 executive order on improving the nation's cybersecurity, the President of the United States highlighted "enhancing software supply chain security" as a vital part of the federal government's efforts [1]. The document addresses utilizing methods of manual or automated testing of software, such as the static and dynamic analysis of source code and use of code review tools. Such testing is a part of the reverse engineering process, in which a thorough understanding of a system's structure, components, and their interrelationships is sought after for the purposes of design recovery or exploit development [2]. Through this capability, the United States can better defend against adversaries and advance offensive tactics in cyberspace.

Reverse engineering (RE) differs significantly from software design and development, or forward engineering. Rather than a defined series of tasks that result in a functioning product, RE is an open-ended process of examination and trial and error, to fully understand what a software system does, how it does it, why it does it, and more [2]. Engineers must often rely on personal experience and general knowledge on artifacts and environmental factors. In a collaborative environment, they may also seek the varied experiences of colleagues. While a team of reverse engineers may benefit from this shared knowledge pool, collaboration in RE poses an additional set of potential challenges.

Improving collaboration can benefit the experiences of new reverse engineers as well. Entry into RE can be daunting for new engineers, as the field is highly technical and there is no systematic training. Because RE is such a niche skill set, there are also fewer RE experts, compared to other software-related fields. Knowledge is most commonly gained on the job by learning from those who are more experienced. Enhancing collaboration practices can impact that process by enabling more efficient flows of information.

Reverse engineering tools were not designed for collaboration [3]. Therefore, when multiple engineers combine efforts on a single file, they must either work on separate copies of the file and compare notes, or take turns working on a shared file. Alternatively, each engineer could be assigned different files in the system to investigate. In either instance, communication and documentation involves external tools. Reverse engineers must employ the collaboration tools that best fit their needs. While studies have explored collaboration in other fields, this research presents the needs unique to collaborative RE and the tools that reverse engineers currently use to meet those needs.

1.2 Research Objectives

Through the experiences of subject matter experts, this research seeks to understand the processes of reverse engineering teams. The results should answer the research question, *How is collaboration currently conducted in reverse engineering?* The primary objectives of the research are as follows:

- Determine the user needs and challenges unique to collaborative RE
- Identify the collaborative tools used by RE teams
- Establish the standard collaborative RE workflow

- Discern how the collaborative tools are integrated into an RE team's workflow

1.3 Research Approach

This research solicits reverse engineers with at least five years of experience to participate in interviews. The participants are asked to recall a critical incident or project in which active collaboration between team members occurred. The survey instrument dives deep into the methods of communication and strategies for documentation throughout the incident. The participants are questioned about the impact of collaborative tools on each step of the collaborative process. The interviews aim to understand the specific needs and challenges that are being met by the tools that the RE teams choose to utilize.

The interview data is analyzed to uncover the general collaborative RE workflow, most commonly used collaborative tools, and their significance to RE teams. Though these tools may not have all been designed specifically for RE, they serve a purpose that reverse engineers identify as critical. Challenges may still exist in the integration of the tools into each individual's workflow. However, recognition of their value to the team must be achieved first.

1.4 Thesis Overview

This chapter offered a brief discussion of the scope and intent of the research. Chapter II reviews existing literature on the individual RE workflow, RE tools, and collaboration solutions in other software-related fields. Chapter III introduces the methodology and analysis strategy employed in conducting this research. Chapter IV provides the results of the data analysis and answers the research question. Chapter V summarizes the research conclusions and offers avenues for future work.

II. Background and Literature Review

2.1 Overview

Chapter I introduced the problem of understanding how reverse engineers collaborate with each other and presented the research question: *How is collaboration currently conducted in reverse engineering?*

This chapter begins with a review of the reverse engineering (RE) workflow and tools used by an individual engineer in preparation for examining the collaborative RE workflow in this research. Next, solutions that have been developed to support collaborative RE are discussed, before exploring how collaboration is optimized in software development. Finally, the chapter concludes with an overview of interviewing methods for eliciting knowledge from subject matter experts.

2.2 Reverse Engineering Process

No standard RE process exists that can adequately capture all the work done by reverse engineers [4]. Every project is unique in its complexity and challenges, and each engineer may take a different approach to complete it. However, the process of reverse engineering an unknown binary is simplified by Quist and Liebrock down to four steps: setting up an isolated environment, executing the program for initial analysis, deobfuscation and disassembly, and finally, identifying and analyzing relevant portions [5]. A similar workflow, but specifically for vulnerability discovery, is identified by Votipka, et al.: information gathering, program understanding, attack surface, exploration, vulnerability recognition, and reporting [6]. These two characterizations of the RE workflow represent the process from a technical perspective. The workflows introduced later in this section will examine the RE process from higher levels of abstraction.

2.2.1 Mental Processes

In a separate study, Votipka, et al. gained insights into the individual reverse engineer's mental processes, using a semi-structured interview protocol [7]. The themes that emerged from the interviews reveal reverse engineering to be an iterative process, depicted in Figure 1.

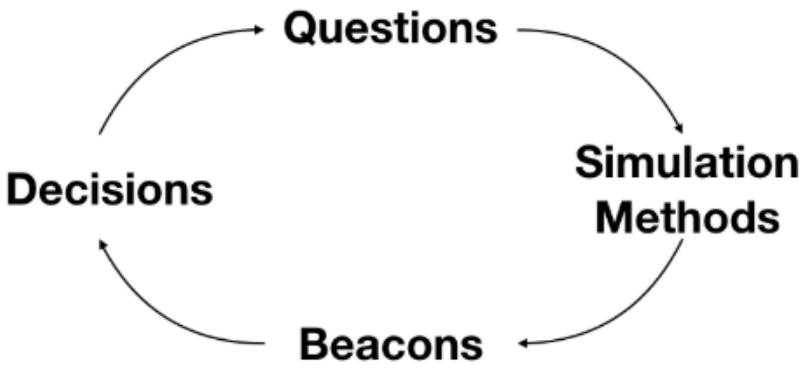

Figure 1: Illustration of the loop reverse engineers iterate through when analyzing programs [7]

"Reverse engineering is inherently a process of knowledge discovery," [7] as high-level goals and questions lead to more specific questions. Once a question is pinpointed, the next step is to simulate or execute a scenario, either mentally or with a dynamic analysis tool. As the program is simulated, the engineer may recognize beacons that point to additional inferences. With those inferences, the engineer makes a decision on how to best use their time, considering further questions and hypotheses to test. These decisions are often made based on prior experience with findings that seem familiar. When there are no similarities to prior work, the engineer makes decisions based on heuristics, such as function size or complexity. The process iterates until the engineer discovers the program's functionality.

2.2.2 Reverse Engineering Workflow

The reverse engineering workflow can be captured from a higher level of abstraction. Based on data from semi-structured interviews with professional reverse engineers, Henry broke the workflow down into five processes [8] illustrated in Figure 2. Metadata analysis includes obtaining the basic, static information about the binary that provides a "broad understanding of the compiled program" [8]. Disassembly analysis involves the use of a disassembler to statically analyze the code. Dynamic analysis most frequently occurs using a debugger in a virtual machine.

The results of the three forms of analysis must then be synthesized and documented in an organized manner [8]. Depending on external stakeholders, the documentation may be translated into language that is able to be easily interpreted by non-technical management. Reverse engineers who are working in teams also meet regularly and share documentation with each other. However, reading one's documentation alone may not be enough to gain an understanding of where they are at in their analysis process. Collaborating in reverse engineering often involves taking

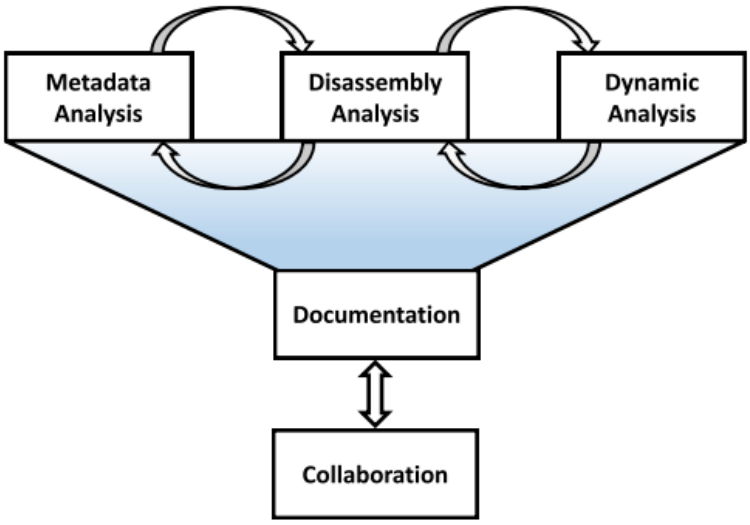

Figure 2: Overview of the reverse engineering workflow process [8]

time to recreate another engineer's work to reach the same level of understanding.

Documentation and collaboration are not commonly discussed as core stages of reverse engineering. However, these stages are as critical to the success of the process as the technical analysis [8]. The challenges reverse engineers face in these stages can translate into preventable inefficiencies, due to the lack of tools specific to the reverse engineer's needs.

2.2.3 Reverse Engineering Workflow in Varying Contexts

A high-level workflow for RE in a security context is identified by Treude et al. as five key processes: analyzing, documenting, transferring knowledge, articulating work, and reporting [4]. Analyzing the assembly code and following the flow of data in a program is "at the heart of most reverse engineering projects" [4]. Documenting serves several purposes, providing cognitive support for the engineers during analysis, while communicating code comprehension to collaborators or outside stakeholders. Transferring knowledge refers to the sharing of completed work beyond what is possible through documentation alone. Treude et al. cite the restrictiveness of workflow support tools in an RE environment as a roadblock to success. Articulating work includes scheduling sub-tasks, recovering from errors, assembling resources, and other items required to coordinate task completion. Finally, the results of the RE activities must be reported when external stakeholders are involved. Breaking the RE workflow in these five processes allowed the researchers to acknowledge the lack of adequate tools to support the flexibility required in RE. The task complexity, security context, and tool constraints of RE "make it impossible to follow a structured heavyweight process" [4].

Wagner, et al. explore the background and problem characterization of malware analysis [9]. They identify the basic differences between static and dynamic analysis

in the process of behavior-based malware recognition. While each approach has its purpose and limits in the reverse engineering process, they both "yield patterns and rules, which are later used for malicious software detection and classification" [9]. Reverse engineers often use static and dynamic analyses simultaneously, "using the results of one to feed the other" [7].

Reports from a series of case studies were analyzed by Ceccato et al. to produce a model that describes how hackers understand an application and identify sensitive assets [10], shown in Figure 3. Static and dynamic program analysis play a dominant role in this process. Beyond enabling code comprehension, they possess the ability to limit the scope of attack. For example, studying string references in the code can reveal "specific constant strings [that are] referenced in the proximity of sensitive assets" [10]. Tampering with execution can also identify sensitive assets, to extract a crypto key for instance. Lastly, when hackers recognize libraries with well-known functionalities, they get important clues on how they function for asset protection. If program analysis and reverse engineering is inhibited and when tampering of program

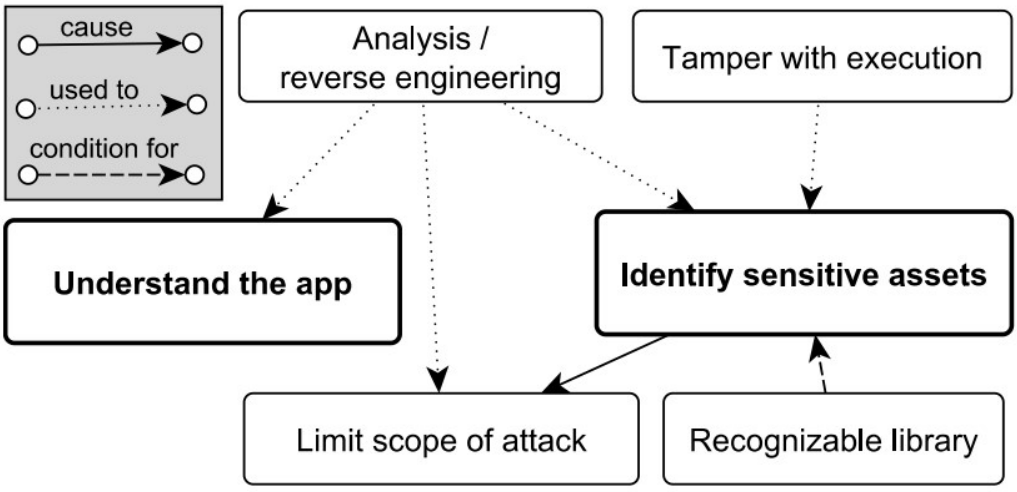

Figure 3: Model of hacker activities related to understanding the app and identifying sensitive assets [10]

execution is not allowed, the hackers' tasks of understanding the application and identifying sensitive assets becomes harder to carry out, as illustrated in this model.

2.3 Reverse Engineering Tools

In the reverse engineering of assembly code, disassemblers and decompilers are semi-automated tools used to analyze software systems and understand the underlying executable program. A disassembler is a program that takes an executable program as input and outputs the equivalent assembly language in a human-readable format. A decompiler takes it a step further and translates the program into an equivalent high-level language, such as C [11].

While performing advanced static analysis, reverse engineers rely heavily on the disassembler program to view the program's functions, libraries, strings, and more [12]. The disassembler can also provide a control flow graph of the program that can help the engineer understand the sequence in which functions are executed and variables change. Upon developing hypotheses and uncovering functionalities, engineers can add comments to the code or rename functions as documentation.

Popular disassemblers also offer a built-in decompiling feature. However, the outputs of decompilers (i.e., the recovered C code) are not extensively used in academia as trusted evidence [13]. The recovered high-level code is commonly believed to be unusable for recompilation, and decompilers are understood to have bugs and deficiencies that may impede the correctness of their outputs. Nevertheless, decompilers remain among the most fundamental tools for code comprehension.

The two most popular reverse engineering tools are Ghidra and IDA Pro. Ghidra is an open source software reverse engineering (SRE) framework developed by the National Security Agency's (NSA) research directorate [14]. Ghidra's capabilities include disassembly, decompilation, graphing and scripting, and more. The tool can

be run in "both user-interactive and automated modes" [14] and supports a variety of processor instruction sets and executable formats. The exposed application programming interface (API) allows users to develop their own plug-in components or scripts. As shown in Figure 4 below, Ghidra's graphic interface displays the assembly code of the selected function in the main window, with side panels that break down the sections of the executable, as well as imports, exports, functions, and strings.

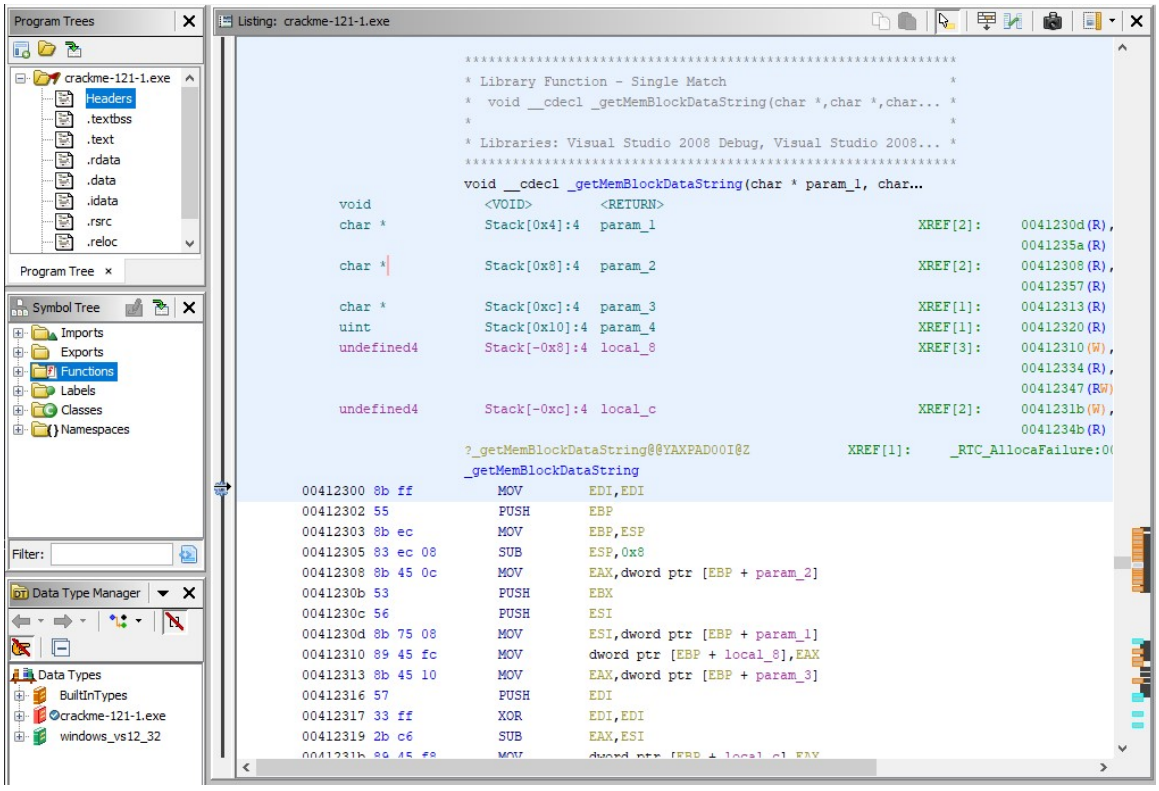

Figure 4: Ghidra disassembler interface

IDA Pro is another widely used disassembler developed by Hex Rays, available for Windows, OS X, and Linux. The tool claims to be able to analyze binaries "in a matter of seconds" [15], and comes with local and remote debugging features. Unlike Ghidra, the free version of IDA does not include select advanced features such as decompiling, Python scripting, batch analysis, and support for processors beyond x86 and x64 [15]. However, the full version supports "more than 65 families of processors

that include x86/x84, ARM/ARM64, MIPS/MIPS 64, etc." [15], a greater number than supported by Ghidra [16]. IDA Pro's interface is similar to that of Ghidra, with the main tab for the assembly code accompanied by a list of the functions. The binary's imports, exports, and strings are also easily accessible as additional tabs.

Binary Ninja is another reversing platform created by Vector 35, featuring a built-in scriptable decompiler [17]. The tool supports the disassembly, lifting, decompilation, inline-assembly editing, and C compilation for a range of processor architectures. While there is no free version of the software, Binary Ninja boasts an extensive API that has enabled an active community of plug-in developers. Binary Ninja's greatest advantage is its modern user interface, that is highly customizable "for a fast, pleasant experience" [17].

2.4 Collaborative Reverse Engineering Tools

This section introduces RE collaboration tools used alongside the general RE tools discussed previously.

2.4.1 Version Control Tools

Ghidra Server is a built-in feature of the Ghidra SRE framework that allows users to create shared project repositories [18]. Shared projects facilitate collaboration among reverse engineers by allowing them to share changes and annotations made to the binaries. With Ghidra Server, users can save a snapshot of a file's current state, also known as a "commit." Collaborators can push or pull commits to and from the server, add comments to each new commit, and maintain a backlog of all commits made to the server.

To create a Ghidra Server shared project, the server must first be set up on any network-connected machine running Ghidra. The server setup files are included in

Ghidra's system files. Given the server IP address and port, anyone can then create a shared project within Ghidra and start adding files. When a file is added to the project, the "Add to Version Control" option will become available in the right-click menu, shown in the leftmost image of Figure 5. Once changes or comments are made in the file and saved, the "Check In" option, shown in the center image, will push those changes to the server for other contributors to view.

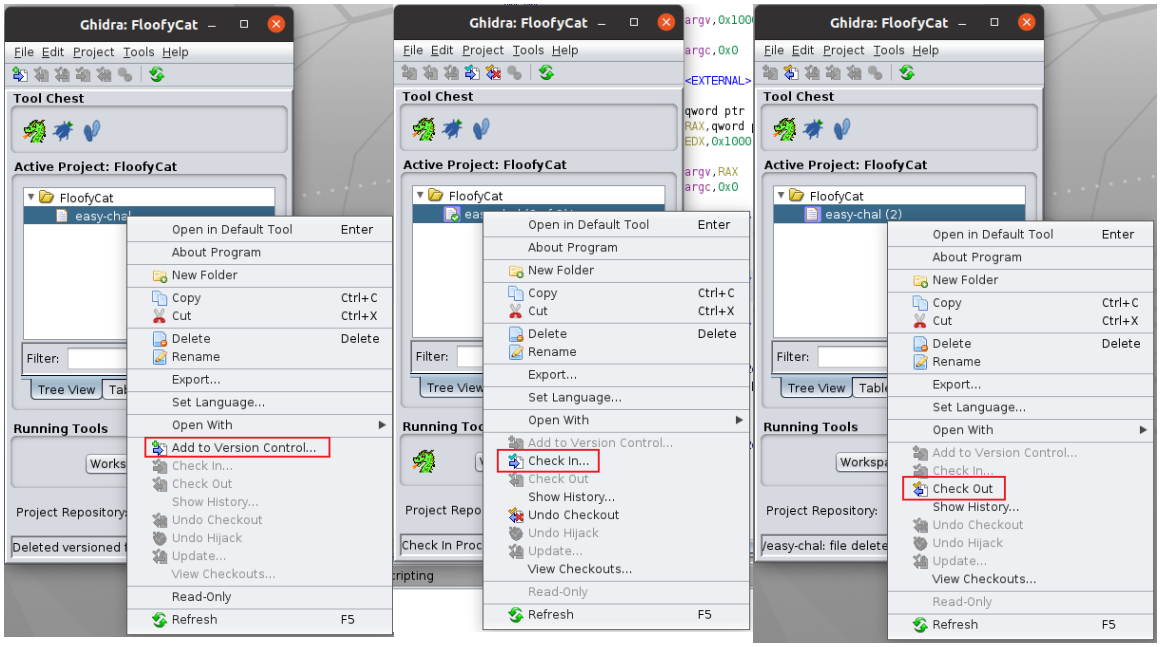

Figure 5: Ghidra server interface

To begin work on a file, users must first "Check Out" the file, shown in the rightmost image. While the file is checked out, if updates were made and checked in by other users, the "Update" option will provide the newest version of the file. A green check indicates that the file is completely up to date. Users can use the "Show History" option to view a complete log of commits, including timestamps, user IDs, and comments. The "View Checkouts" option also shows who has opened the shared file, even if no changes were made and pushed. These capabilities allow teams to maintain documentation of the work done on a project involving multiple reverse

engineers.

For teams that use a variety of disassembling tools, CollaRE allows teams to share project files from separate locations. The tool does not integrate directly into any RE tool, but supports project files from Binary Ninja, Cutter, Ghidra, Hopper, IDA, and JEB [19]. Simple version control, including check-in and check-out operations, manages files produced by these tools so that all members of a team have access to work done by one another.

2.4.2 Workflow Synchronization Tools

No other disassemblers have a shared project feature like Ghidra Server built into the program. However, third party developers have created plug-ins for IDA Pro and Binary Ninja that support collaboration. One example is CollabREate, an IDA plug-in that goes one step beyond sharing files on a server. CollabREate also captures user actions and database updates. Actions that are captured include function added, byte patched, operand type changed, comment changed, and more. Users can selectively choose to publish this data to the server, subscribe to changes that others publish, or both publish and subscribe [3]. Users have "granular and flexible control" [3] at the project level or individual session level over what is shared. For example, a user can choose to only publish comments and subscribe to patched bytes and function name changes. This flexibility is particularly useful if engineers have varying levels of experience or play different roles on the team, but need to keep their materials synchronized.

IDArling is another IDA plug-in that supports collaboration by synchronizing "real-time changes made to an IDA database by multiple IDA users" [20]. The plug-in collects one user's actions within IDA, just like CollabREate, but immediately propagates the changes to other users that have the same snapshot loaded, through

the server architecture. While some teams may prefer the granular control over what is shared that CollabREate offers, others may value the efficiency that IDArling's real-time synchronization affords instead.

Visualization needs in collaborative RE are addressed by SensorRE, a plug-in for Binary Ninja that leverages the concept of analytic provenance. Analytic provenance is the understanding of "a user's reasoning process through the study of their interactions with a visualization" [22]. SensorRE captures user actions in Binary Ninja as they are executed and provides a provenance graph that visually communicates a timeline of events. The provenance graph, depicted in Figure 6, is a vertical tree made up of nodes, edges, and branches. Each node represents a state in the disassembler.

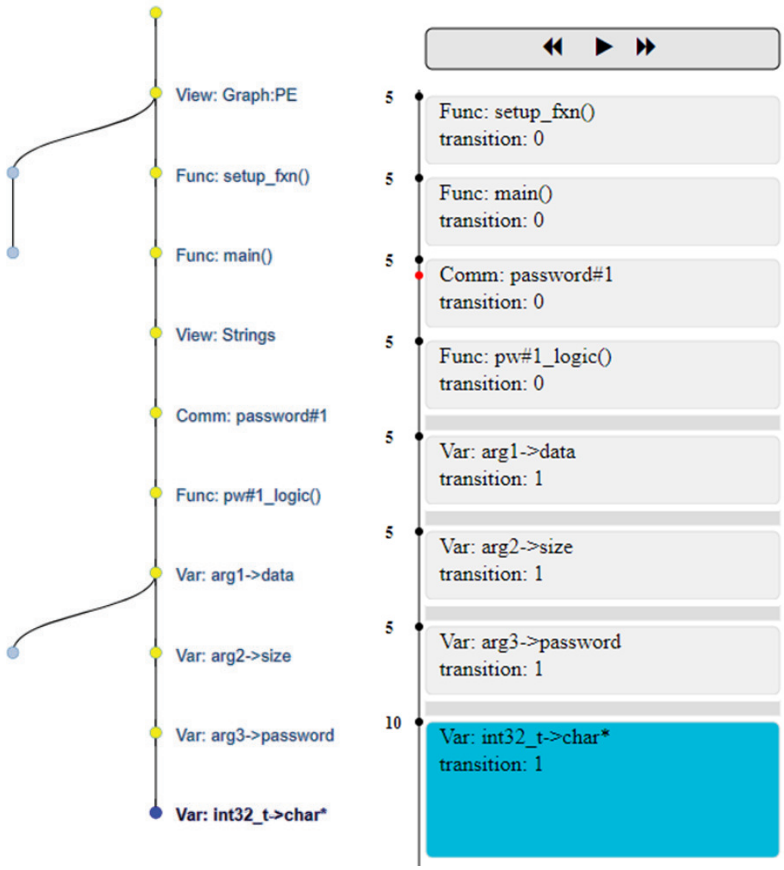

Figure 6: SensorRE visualization with a captured provenance graph and storyboard [21]

Each edge represents the user action taken that resulted in a state change. Branches occur in the tree where the user tried different courses of action to test potential hypotheses. This scalable visual history of actions can be replayed sequentially using the storyboard side panel of SensorRE, also shown in Figure 6. The graph and storyboard can be exported and presented to other engineers or project stakeholders to describe the user's workflow.

2.5 Collaboration in Software Development

Compensating for communication loss in geographically distributed teams has driven research and design of collaboration tools for software development, or forward engineering. Christian and Rotenstreich created an evaluation framework for such tools that represent the needs of asynchronous teams [23]. Through an investigation of tools that have proven beneficial to asynchronous teams, five key characteristics emerged: awareness, calendar assist, context persistence, coordination, and visualization. Awareness tells the user who is in the workspace and what they are working on. Calendar assist refers to the ability to coordinate the team to conform to a schedule and notify them of meetings and deadlines. Context persistence allows users to see events that have occurred, as well as who or what contributed to them. Coordination mechanisms includes rules and procedures that facilitate the prompt completion of distributed tasks. Visualization provides a visual channel for large amounts of data and information exchange. When evaluating a collaboration tool, the more of these five characteristics the tool exhibits, the more effective it is at supporting a distributed team.

Seven standard classes of collaborative development tools were identified by Lanubile, et al.: version-control systems, trackers, build tools, modelers, knowledge centers, communication tools, and Web 2.0 applications [24]. Version control systems, such

as Git and Mercurial, allow developers to share software artifacts and maintain team synchronization. Trackers, such as Jira, help a team manage and document issue resolutions in an ongoing project. Build tools offer secure, remote repositories and build management. Modelers help developers create artifacts such as Unified Modeling Language (UML) models and customized software processes. Knowledge centers contain internal documents, technical guides, standards, and best practices for team members to share with each other. Communication tools can support both asynchronous and synchronous communication, with features ranging from videoconferencing and mailing lists to document sharing and concurrent editing. Web 2.0 applications, such as Confluence, allow direct user contributions and community building, representing a valuable means to increase overall communication among team members. Supporting all the types of collaborative tools has become a strategic priority for teams with distributed resources and software needs.

As discussed previously, version control systems are integral tools in collaborative software development. Git is the leading software in this category, as its functionalities and efficiency have evolved by leaps and bounds since its creation in 2005 [25]. Just like its predecessors, Git tracks changes in working source code and allows multiple developers to work together. However, Git's unique branching model sets it apart from other version control systems. If a repository's history of revisions is to be thought of as a linear tree, Git users are encouraged to create local "branches" in the tree that remain independent of each other. These branches can be deleted without impacting the rest of the tree, or merged with the main line [26]. Users can seamlessly switch between different branches within a repository, experimenting with different feature ideas without fear of breaking existing features.

Git is also a distributed control system. Being distributed means that each developer in the team has their own copy or "clone" of the repository. Therefore, if the

team is using a centralized workflow, depicted in Figure 7, each developer has a full backup of the files.

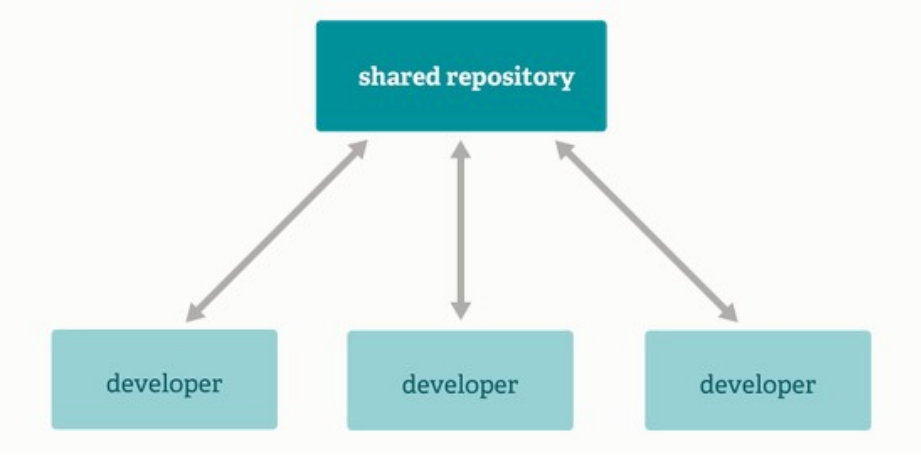

Figure 7: Subversion-style workflow [27]

Git's distributed nature, combined with the branching system, enables easy implementation of a wide range of workflows. For instance, a common approach involves an integration manager, who is the sole person who can commit to the "blessed" repository. Developers must clone from that repository, push to their own independent repositories, and then ask the integration manager to pull in their changes [27]. This workflow model, shown in Figure 8, is commonly seen in open source projects.

Microsoft's Visual Studio integrated development environment (IDE) is a code

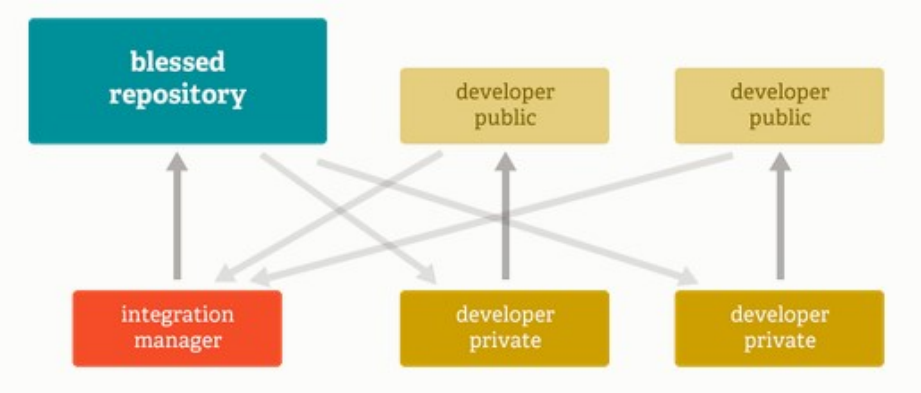

Figure 8: Integration manager workflow [27]

development tool that offers the Live Share extension feature for collaboration. In a Live Share session, multiple Visual Studio users can "co-edit, co-debug, chat with peers, share terminals, servers, and look at comments" [28], all in real-time. Live Share does not require users to clone a repository or configure an environment to function. Inviting collaborators is as simple as sharing a link. Users can be working from any operating system and with any language. Tools similar to Live Share have been created by third party developers as standalone collaboration tools or for use with other IDEs such as Atom [29]. However, Live Share is the only collaboration functionality of its kind that is offered by the creator of the IDE. Microsoft's code editor, Visual Studio Code, also supports the Live Share extension.

2.6 Interviewing Methods

As the director of the Defense Advanced Research Projects Agency (DARPA) from 1975-1977, Dr. George H. Heilmeier crafted a set of questions to help agency officials think through and evaluate proposed research programs [30]. They became known as the "Heilmeier Catechism" questions. When proposing a new research idea, the questions ask the researcher to reflect on ideas including, "how is it done today, what are the limits of current practices, and what difference will it make if you are successful?" In this research, select Heilmeier Catechism questions are used to ask the participants about the state of collaborative tools currently employed in RE.

The Critical Decision Method (CDM) is a semi-structured retrospective interview protocol aimed at capturing expert decision making in a naturalistic environment [31][32]. The technique has proven useful for eliciting expert knowledge, decision strategies, and cues in cognitive research. In a CDM interview, the subject matter expert is asked to identify a specific incident of interest. The incident is then dissected in four stages:

1. Incident selection and recall
2. Timeline verification and decision point identification
3. Deepening
4. "What-if" queries

Votipka, et al. implemented a modified version of the CDM in their study's interview protocol [7]. Participants were asked to choose an interesting program they recently reverse engineered and demonstrate the steps they took. Throughout the program walkthrough, the interviewers asked directed questions to probe the participants' mental models. The researchers used an iterative open coding method to identify themes from the interview transcripts.

A 2003 study investigating the factors behind surgical errors in hospitals also utilized the CDM to elicit details from relevant incidents [33]. Surgeons were asked to provide an open-ended description of each incident and the factors that contributed to the errors made. The interviewer and surgeon then discussed the events of the incident in detail. Finally, the interviewer asked the surgeons about the role of 15 predetermined possible contributing factors. This procedure follows the basic structure of the CDM while tailoring the "what-if" queries stage to the unique objectives of the study.

In a study on bar coding in medication administration and improving patient safety, the researchers conducted CDM interviews with nurses and pharmacists to understand "near-miss, wrong-patient" incidents involving bar code technology [34]. By recognizing recurring patterns across the CDM interviews, the researchers were able to identify unanticipated side effects of introducing the technology to existing practices.

Similarly, in this research, the CDM is used to understand the existing practices and workflow of collaborative RE, as well as the reasoning behind the choices of tools

utilized. This method offers the researcher the most structured and consistent way to collect the data required to answer the research questions. The specifics of the implementation of the CDM are detailed further in the research's methodology.

2.7 Chapter Summary

This chapter reviewed how RE is conducted by individuals and the tools required. The existing solutions for collaborative RE are discussed, along with collaborative tools used in software development. Finally, relevant interviewing methods are introduced ahead of the next chapter's description of the study's design.

III. Methodology

3.1 Overview

The purpose of this research is to discover the needs and challenges of a reverse engineering (RE) team and the collaborative tools implemented to meet those needs. Selecting the appropriate methodology is critical to fully understand how RE teams collaborate. Figure 9 provides an outline of the method followed. The process starts by defining the research problem, selecting the suitable research method, and designing the study. After the design is executed, data analysis is performed to present results that answer the research questions.

This chapter describes the methodology used to collect and analyze the study's data and the rationale behind the chosen approach. The discussion includes the design of the interview process and questionnaire, how participants were identified and selected, and the data analysis strategy.

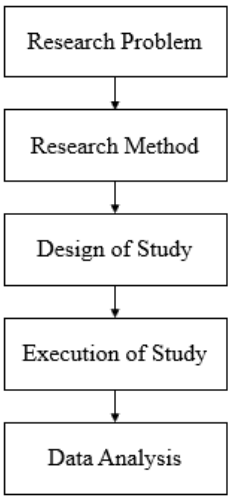

Figure 9: Overview of the research methodology

3.2 Research Problem

RE is conducted in teams to increase the rate at which software systems can be dissected or exploits can be developed. Adding engineers may allow the work to be split up and given multiple perspectives. However, increasing the number of personnel does not always directly translate into greater efficiency. At a minimum, a team must have the proper communication channels and documentation practices in place to see increased productivity. If a team does not make these considerations, potential challenges may include redundancies, inconsistencies, or misunderstandings. Multiple engineers may end up unintentionally analyzing the same function, wasting time that could be spent making discoveries elsewhere. Team leads may have difficulties keeping track of who is responsible for the completion of assigned tasks and holding members accountable for their work. Without an organized method of documentation, individual engineers may spend extra time searching for archived material that they need on a current project. These hypothetical dilemmas represent how collaborative RE differs from individual RE, and require more than a simple conversation to address.

Obtaining a comprehensive understanding of the collaborative RE environment and existing workflow is necessary to identify solutions to meet the specific needs of the users. The question driving this research is: *How is collaboration currently conducted in reverse engineering?* Several supporting questions were developed to support this core question:

1. What does the collaborative RE workflow look like, from start to finish?
2. What are the user needs and challenges unique to the collaborative process?
3. What collaborative tools do RE teams currently use?
4. How are those collaborative tools integrated into a team's workflow?

3.3 Research Method Selection

There are two primary methods for conducting research studies, quantitative and qualitative. Quantitative approaches have features advantageous for researchers looking to make statistical generalizations on a large number of cases. They use standard measures to fit many perspectives and experiences into predetermined responses [35]. However, without utilizing statistics, qualitative studies enable the researcher to gain more depth and detail on a limited number of cases, while allowing subjectivity of individual experiences to be captured. For this research, a qualitative study was chosen to seek a variety of experiences of subject matter experts (SMEs).

A series of semi-structured interviews with reverse engineering SMEs were conducted in accordance with the interviewing strategies outlined by Wood. The goal of Wood's techniques is to describe users' current work practices and propose ways their work can be enhanced by introducing a software support application [36]. The SMEs have the professional experience necessary to credibly speak on the goals and challenges of reverse engineering in a collaborative environment. Their knowledge and experience with collaborative tools can also help reveal how RE teams choose to take on those challenges, addressing the research questions.

3.4 Design of Study

The semi-structured interviews follow a predetermined format in which participants are asked the same questions in order. However, depending on the participant's responses, further elaboration, clarification, or examples may be necessary [37] [38]. Semi-structured interviews allow the interviewer to adapt to the conversation and elicit more intricate details from the subject while staying on theme throughout the allotted time. This section describes the purpose of each group of questions and how the subject matter experts were selected.

3.4.1 Purpose of the Question Groups

Before beginning the questionnaire, the researcher read the purpose of the study aloud for the participant's awareness, and asked if there were any questions or concerns before starting. This standard introduction reduces the likelihood of misunderstandings or off-topic responses from the participant throughout the interview.

The preliminary interview questions serve as a demographic survey of the participant. Without requiring personally identifiable information, the questions ask about the participant's educational background, amount of professional reverse engineering experience, and nature of that experience. Though demographic data is not a primary variable of interest, the information could be referenced during data analysis to explain trends or outliers.

The remainder of the interview is organized into two lines of questioning. The first section prompts the participants for initial thoughts on the state of collaboration tools and techniques available today to reverse engineers, including their history and evolution, as well as benefits and limitations. These questions are inspired by the Heilmeier Catechism questions [30], because they effectively establish current best practices and shape the problem space for future developments.

The second line of questioning follows the Critical Decision Method (CDM) of eliciting knowledge. The CDM offers the ability to capture tasks in *"naturalistic environments characterized by high time pressure, high information content, and changing conditions" [37].* Figure 10 depicts the CDM interview process in four sweeps of questioning [32]. First, the researcher asks the participant to recall a specific, past incident in which they collaborated with others to accomplish a RE task or project. Because this prompt immediately follows the Heilmeier Catechism questions, the participants will likely bring up incidents relevant to the limitations of tools discussed previously. Based on the participant's overview of the incident, the researcher creates

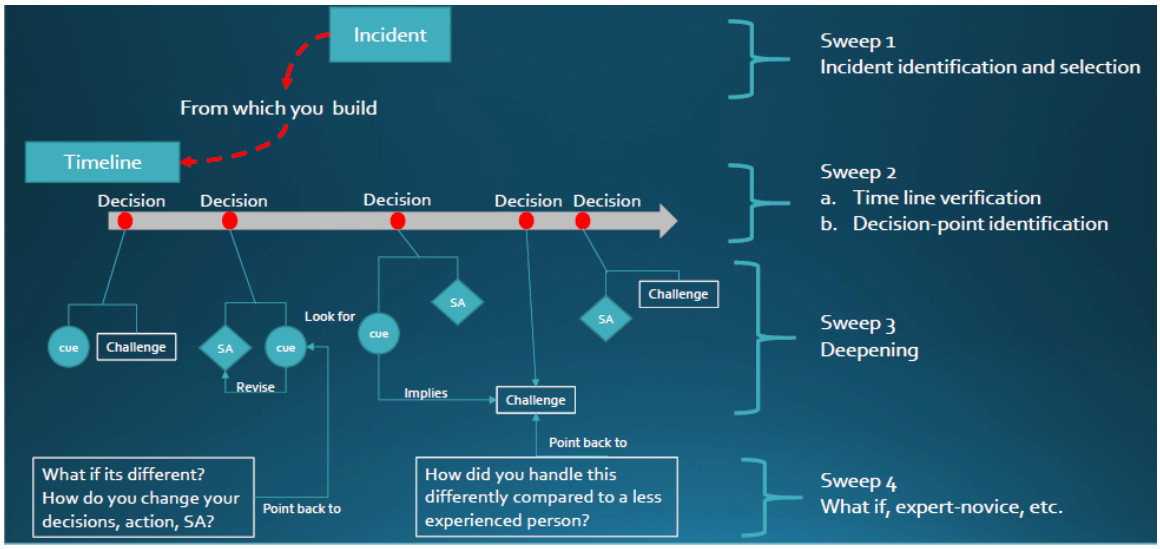

Figure 10: Overview of the Critical Decision Method [37]

a timeline of events and identifies key decision points in this timeline. After verifying the timeline's accuracy with the participant, the researcher begins the next sweep into each decision point.

In the ensuing deep dive into each point on the timeline, the participant identifies the specific challenge encountered at, or prior to, that decision point. The participant is encouraged to expand and reflect on the team's use of collaborative tools during the time of the incident. For each challenge, the researcher asks the following follow-up questions:

1. How did the involved engineers' level of experience (at the time) contribute to the handling of the situation?

2. How did the integration of collaborative tools affect the courses of action taken?

3. From this decision point, what were some lessons learned? Did any of these lessons reappear later in the project?

After exploring every key decision point, the researcher restates all of the lessons learned that the participant previously mentioned. "Did any of these lessons have an impact on a future collaborative RE process?" The lessons that the participant

reiterates in response to this question are more likely to be of greater significance than others mentioned. The researcher also collects ideas for possible improvements to available tools, as well as possible risks, trade-offs, and indicators of success for such changes.

Finally, the researcher offers the participant an opportunity to provide any additional thoughts or closing comments on any subject that was not previously mentioned. The complete questionnaire can be found in Appendix C.

3.4.2 Pilot-Testing the Interview Questionnaire

Before approaching potential study participants and proceeding with the interviews, the researcher submitted the survey instrument to the Institutional Review Board (IRB) for approval, documented in Appendix A. Once approved by the IRB, the researcher proceeded with a test run of the questionnaire with a reverse engineer of intermediate experience. The purpose of the test run was to ensure that the elicited responses remained relevant to the research question and did not confuse the participant. An intermediate-level reverse engineer was selected because if they have enough experience to draw meaningful thoughts from, an engineer with greater experience would likely be able to as well. The results of the test run demonstrated that the questions were appropriately structured and relevant to the research problem.

3.4.3 Selection of Participants

Research participants were selected based on their knowledge and experience in software RE in collaborative environments. Participants have at least five years of experience with software RE in their professional careers. Recruited reverse engineers have also collaborated with others to complete RE projects in the past. Participants were solicited from active duty military members, Department of Defense

civilians, and contractors. The researcher recruited them from U.S. Air Force units that conduct software reverse engineering on a day-to-day basis. The participants were contacted by the researcher through email and informed that there would not be compensation for their participation.

3.5 Administration of Interviews

The interviews were conducted throughout November 2021. Participants were given the option of an in-person or phone interview. Four participants opted for a call, while the other four met with the researcher in-person. The length of the interviews ranged from 50 minutes to an hour and 40 minutes. The researcher took notes during the interview in a text editing software. The dialogue was also recorded using voice recording software that distorted the incoming audio to mask the participant's identity. Text and audio file names did not include any personal identifiable information for the same reason. Throughout the interview, the participants were given as much time as they required to answer the questions to their satisfaction.

The researcher took note of technical terminology and asked for clarification where necessary. In each interview, the researcher was careful not to mention new terminology learned from previous interviews, to avoid data contamination across interviews. The researcher avoided asking leading questions that alluded to the responses of other participants or any preconceived response.

3.6 Data Analysis

Upon completion of the interviews, the researcher transcribed the audio data and compiled the text into a readable narrative for analysis. Qualitative analysis connects, describes, and classifies the data collected to produce meaningful themes, patterns, and insights [35]. To achieve this, the interview data and the researcher's

notes were loaded into the data analysis software, MaxQDA [39], for easy viewing and comparison. Analysis highlights are easily retrievable in MaxQDA, with a range of filter options displaying all relevant data and results.

Even though the CDM approach to interviewing yields a unique scenario for each participant, similarities still emerge in the challenges and decision points encountered. In MaxQDA, the researcher could associate blocks of text with major themes and assign labels and weights to the text segments. Themes and patterns are acknowledged and reported in the following results.

An example of how the responses produce patterns across participants is described. The questions concerning current collaboration tools and techniques asked in the questionnaire are, "Describe the tools/techniques utilized today while performing RE with others, specific to the collaborative environment. If they have evolved over the course of your career, please specify how," and "What are the benefits and limitations of the tools/techniques used today for collaboration in RE?" After consolidating all the responses on current tools and techniques, the researcher found that the majority of experts brought up the Ghidra Server plug-in for shared version control. The high frequency of these responses indicate that the use of Ghidra Server is prevalent, rather than a one-time phenomenon.

3.7 Chapter Summary

This chapter introduced the method for conducting the study and the research questions that drove that choice. The interview questionnaire was created to guide the deep dive into the participants' experience with collaborative RE. Interviews were conducted across the span of three weeks with participants from a variety of professional backgrounds. The data collected from the interviews were sorted and analyzed to yield answers to the research questions.

The next chapter presents the findings of the semi-structured interviews with the subject matter experts. The interview data is explored to understand the tools utilized and the needs they address in a collaborative RE environment.

IV. Results and Analysis

4.1 Overview

This chapter presents the data collected from the interviews introduced previously. The study's participants come from a variety of educational and professional backgrounds and provide insights from their experiences. The interview data assists in establishing the collaborative reverse engineering (RE) workflow and identifying challenges and user needs. The participants also discuss the collaborative tools currently employed and how they fit their needs. This chapter addresses the primary research question: *How is collaboration currently conducted in reverse engineering?*

4.2 Interview Data

Eight subject matter experts participated in the study, listed in Table 1. Each participant has been a part of software RE teams with the purposes of either design recovery or exploitation development. Educational backgrounds are limited to bachelor's or master's degrees in computer science, computer engineering, or electrical engineering. Half of the participants have had only 5 years of professional RE experience *(P1, P3, P5, P6)*, while the other half have had 10+ years of experience *(P2,*

Table 1: Participant demographic survey

Participant	Degree	Education	RE Experience
P1	Comp Sci	B.S.	5 years
P2	EE	M.S.	11 years
P3	Comp Sci	M.S.	5 years
P4	Comp Sci	M.S.	12 years
P5	Comp Sci	M.S.	5 years
P6	Comp Eng	M.S.	5 years
P7	EE	B.S.	10 years
P8	Comp Sci	M.S.	11 years

P4, P7, P8). Two of the more experienced participants served as their team's lead engineer on at least one project *(P4, P8).* All participants reported receiving no RE-specific training beyond basic assembly language taught in undergraduate classes. Advanced RE knowledge was learned solely on the job, working under those with greater experience.

4.3 Collaborative Reverse Engineering Workflow

During the CDM portion of the interview, the participants outlined the general workflow followed by a team they worked on, derived from the memory of a specific, past RE project. After synthesizing all eight responses, the collaborative RE workflow can be generalized as shown in Figure 11.

Upon receiving a new project, the team lead performs a brief first pass of the system, taking inventory of all files and determining its basic structure. If provided a

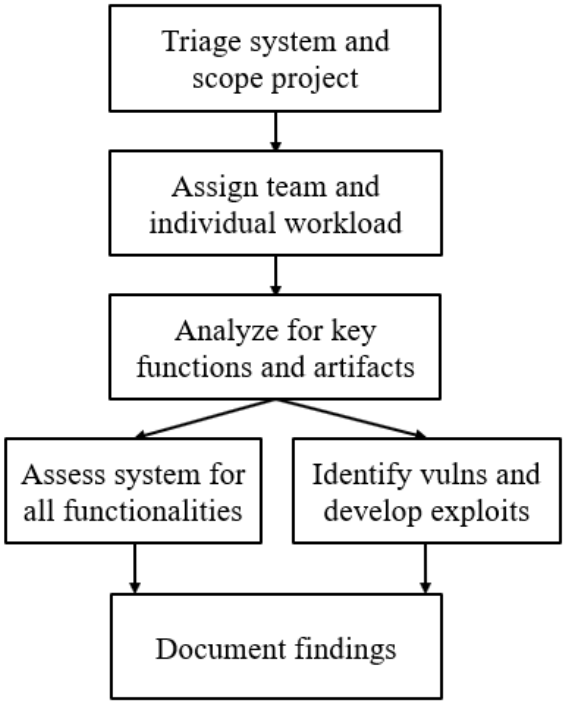

Figure 11: Collaborative reverse engineering workflow

previous high-level report by the client, they would verify the background information included to get an accurate picture of the architecture and design of the system. This initial process identifies the resources that the team needs to build an appropriate virtual environment around the system for it to *"reside in its most natural state" (P4).* This process is known as "rehosting." Any intelligence on the system's origin or attribution could also prepare the team for what kind of tactics, techniques, and procedures to expect moving forward.

The size of the team assigned to any given project is highly dependant on the complexity of the job, as well as the time frame they are given to complete it. *"The largest project I've ever worked on had a team of 12, all hands on deck, because it was on a very hot topic at the time" (P5).* Most large projects require up to four or five engineers. However, an average project typically warrants no more than three engineers, including the team lead *(P4, P5, P7).*

Next, the team lead starts assigning various tasks or components of the system to the team members. Some reverse engineers may have expertise or past experience with particular architectures and types of binaries. Others may have more limited skill sets and experience at the time. The lead takes everyone's abilities into account and plays to the engineers' strengths as much as possible to maximize the team's work efficiency *(P2).* Team members may also voice a preference for working on a particular part of the system. These requests are often granted, as one is more likely to stay motivated working on something they have greater interest in *(P5).*

The individual engineer starts a rudimentary analysis of their piece of the software, looking for functions that stand out, interactions with other files or systems, or artifacts such as IP addresses, encryption keys, and file names. This process often involves consulting other team members to share hypotheses, identifying common patterns, or simply asking for help when challenges arise. Maintaining open lines of

communication optimizes the team's coverage and thoroughness. For example, if one engineer stumbles upon an IP address while working on the encryption protocol, they may pass it on to someone who is working on network artifacts *(P3)*. This shared observation could either bring new awareness to the address and its significance or confirm an existing hypothesis. Collaborative RE also comes with the inherent benefit of an expanded knowledge pool. For instance, when faced with an unfamiliar binary packing technique, another engineer may have encountered said technique in a past project and successfully unpacked the file then. Reaching out to each other when in unfamiliar territory can save the engineers valuable time and allow them to learn something new in the process *(P3)*.

Constant communication can not only facilitate productivity and prevent redundancies, but save each other headaches from *"going down endless rabbit holes" (P2). "Reverse engineering is agile, things don't always turn out to be what you thought, and before you know it you find yourself chasing shiny objects" (P4).* This phenomenon can happen to the most experienced of reverse engineers, and there is no solution guaranteed to help avoid it entirely. However, an increased awareness of what others have worked on and found can often help an engineer determine what is or is not worth taking an deeper look at.

The engineers' next move depends on the mission of the team, as shown in Figure 11. Of the eight participants, five were on teams that primarily worked on exploit development, while the other three focused on design recovery. If the mission is design recovery, the goal is to achieve a thorough understanding of each component of the system. This end state may include a full list of system capabilities, command and control (C2) communication mechanisms, and the big-picture intent of the software. If the software is confirmed to be malicious, the team should also develop a mitigation plan to detect and eradicate it from host machines and prevent reentry

(P3). During this stage, the engineers are still making key discoveries individually. But open discussions and constant updates can aid the team in connecting the dots from everyone's work to paint a complete picture.

Alternatively, if the team's primary objective is exploit development, the engineers are only looking to understand the system's functionality to the extent required to identify vulnerabilities and create an effective exploit. This level of understanding is dependant on the exploit, its access vectors, and potential unintended consequences *(P1)*. Regardless, the team's work efficiency benefits from keeping everyone on the same page, sharing objectives and courses of action, and maintaining situational awareness.

After the engineers complete their analyses or exploits, the team must put together a report that documents their findings with technical details to support. *"Assuming everyone found a vulnerability, individual team members are responsible for writing up their own, which often makes the report's writing style inconsistent and it takes some time to get it all to flow" (P5).* One drafted document template is passed around to the engineers, with areas sectioned off to fill in their unique analyses. In the case of a design recovery report, the entirety of the program's address space is tackled, describing, *"what address space is responsible for what tasks and identify cross-referenced addresses and such" (P5).* Though executive level summaries are included, these technical reports are intended for offensive and defensive operators to read *(P4)*. Oftentimes, engineers can directly paste details from their own notes or group repository to demonstrate how conclusions were reached.

4.4 User Needs and Challenges

Analysis of the interview data reveals common challenges that reverse engineers face while collaborating. The key challenges include maintaining documentation,

mitigating redundancies, supporting continuity, lack of version control, efficient group communication, and interpersonal factors. This section explores how these problems can emerge in a collaborative RE environment and the needs that reverse engineers require to overcome each challenge.

Regardless of the nature of the project, the interviewed subjects' teams worked in close physical proximity. This setup allowed the engineers to verbally communicate and see what others were working on at any given time. When an engineer encounters roadblocks and consults another team member, *"shoulder surfing is the most common way to collaborate and solve problems" (P6).* If this colleague has previous experience with the issue, they can verbally guide the engineer through the solution or point them in the right direction within minutes. In other instances, sometimes it just requires a set of fresh eyes to point out what the engineer may be overlooking. This means of collaboration is simple and effective. However, it relies on the conscious effort of the engineer to leave behind any documentation of what transpired. If a similar problem were to arise again later on, having a documented solution to refer to would be beneficial, rather than taking an individual away from their work again to help.

When multiple people work on the same system with the same objectives, RE teams inevitably encounter redundancies in their work, without proper communication. Though in certain instances, having a second set of eyes on the same problem could be beneficial, *"nine out of ten times, it's unnecessary and you'd be better off without. Intentionally placing two people on the same issue tends to be just a waste of time and effort that could be spent getting something else done" (P2).* Teams err on the side of having insufficient manpower for all the required tasks, rather than an excess. Exceptions occur where there is a major discovery fundamental to a new understanding of the system that requires validation.

A team may be able to avoid unintentional redundancies with verbal updates

and periodic team meetings. However, they would benefit further from a sustained method of sharing progress and viewing each other's past and planned work.

> *"It's really great to be able to see what others have done, are working on, and plan to work on at any given time. If you need to reference someone else's work because it interacts with your stuff, that information is there. If you want to know if a possible rabbit hole has already been looked at, you can check and save yourself the trouble" (P4).*

Such documentation can serve as a central hub for the team, minimizing overhead and streamlining the flow of critical information.

The timeline of an average RE project worked on by three to five engineers can span approximately six months. During this time, the team composition is subject to change and ongoing work may be passed on to new team members. Maintaining continuity is a common challenge, since insufficient documentation and communication often hinders a seamless transition. If possible, the incoming and outgoing engineers sit down together and go over all the discoveries and observations made up to this point, as well as possible next moves and objectives. However, once left on their own, the remaining engineer may find themselves spending more time playing catch-up than progressing in the project. Deciphering others' old notes is often easier said than done, as every engineer has their own method and style *(P2)*. Time can also be wasted recreating past analyses to understand how conclusions were reached. A shared working repository for notes can standardize how this information is organized and stored, if the engineers can get into the habit of tidying their documentation in tandem with making advances.

Though the engineers are each assigned their own portion of the software system to analyze, the team should remain aware of what version of the files they are all looking at. Circumstances change and new revelations happen regularly. Manually redistributing all the project files every time a binary is successfully unpacked or

decrypted would be inefficient and cumbersome to the team's workflow, inevitably resulting in errors. *"Built into a disassembler [the team] already uses, it's nice to be able to maintain version control of the binaries, including comments and renamed functions, just like you would with source code in forward programming" (P6).* Centralizing documentation is pointless if the base material is inconsistent and cannot be referenced reliably.

When consulting each other for assistance or inspiration, engineers commonly default to verbal communication for convenience and quick response time. *"There is tremendous value in informally talking with the people around you, helping each other pick up on things that might otherwise go unnoticed" (P4).* While the documentation of such verbal interactions has been discussed previously, there are alternative methods of communication that serve different purposes. For instance, an engineer may stumble upon an IP address and want to reach out to the team to see if it has been identified elsewhere in the system. Bringing this inquiry up verbally may not be the most appropriate means. Not only are technical information like IP addresses and hashes cumbersome to relay verbally, other team members may miss it entirely if they are immersed in their own work at the time. A group chat program allows the team to simultaneously get word out quickly, allow for a flexible turnaround, and maintain records. If another engineer did not recognize the IP address at that time but discovers it later on, they can go back through the chat log and get in contact with the one who initially brought it up.

In any project, collaboration cannot be neglected when there is a significant skill disparity across the team. Both experienced and newer reverse engineers can make choices in communication that can affect team productivity. Because expertise in reverse engineering is developed almost entirely on the job rather than in a classroom, rookies almost always come in as a blank slate. At this stage, *"it is so important to*

be vocal about what you don't understand and be asking questions" (P4). Many less experienced engineers can be reluctant to reach out for help, resulting in nothing getting completed. This hesitance can be attributed to personality traits or fear of looking incompetent. Though not always the case, junior engineers also tend to be less likely to champion for the ideas they have *(P2).* Meanwhile, more experienced reverse engineers naturally possess more knowledge and insightful tips, given the greater amount and variety of software and situations encountered in their careers. However, *"it isn't uncommon to hear about some of the [veteran engineers] refusing to share what they know or are experts in" (P5).* They may feel a sense of irreplaceability and pride in their skillsets. This choice can have both individual impact, impeding *"opportunities for mentorship" (P2)*, and a greater impact on the workplace for the whole team. Open communication regarding the expectations on team members of all levels of experience would help to make the team comfortable conveying their own needs. Mutual understanding and *"being connected well on a personal level affects the work they bring to the table" (P5).*

> *"When people feel like they are contributing, there is more motivation, productivity, and efficiency. When people feel like they're fighting against the team, there is less work getting done" (P7).*

The need for communication does not end once it comes time to write up the project report. While being passed around from one engineer to the next, the report undergoes a multi-layered review process by the team lead *(P4, P5).* If sections of the report are inadequate, the lead might sit down with the engineer responsible and ask them to explain the concept they tried to document. Through this conversation, either the engineer realizes what was missing, or the lead is able to give better feedback on how to improve the report *(P2). "Sometimes something thought to be common knowledge isn't. [The engineer] needs to make it clear to the reader how they got from*

A to B [with] actionable and relevant notes" (P4). Other times, an engineer may *"make things harder for themselves and the reader with just words" (P5)* instead of using a diagram or table to better convey the information, until prompted.

4.5 Collaborative Tools Employed

For each of the needs and challenges discussed in the previous section, a tool built for workplace collaboration has aided RE teams in its own unique way. Though complete integration of the tools into existing workflows remains a challenge, their value and necessity are acknowledged by both experienced and new reverse engineers. Each column in Table 2 represents a collaborative tool, while the rows list the user

Table 2: Collaborative reverse engineering user needs and solutions matrix

	Issue Tracking Software	Shared Workspace/ Wiki	Team Chat Platform	Ghidra Server
Documentation of Individual Work	✓	✓		✓
Documentation of Team Communication	✓	✓	✓	
Mitigating Redundancies in Concurrent Work	✓	✓	✓	✓
Continuity Documentation	✓	✓		✓
Team Messaging			✓	
Project Version Control				✓

needs that the tools meet. The tools identified by the experts interviewed include issue tracking software, shared workspaces, team chat platforms, and version control systems.

4.5.1 Issue Tracking Software

Issue tracking software provides an elegant solution for the organization of tasks. Within a team's workspace, members are assigned projects which require a series of tasks. Tasks are commonly categorized as completed, ongoing, or to-do, using labels *(P3)*. Each task is a new issue ticket, containing details such as assigned personnel, timestamps, comments, and subtasks. These features facilitate the documentation of every course of action throughout an RE project, starting at the beginning. *"Once [the team] has reached an initial understanding of the all tasks that need to be completed, [the lead] can create them and start assigning people to them all within Jira" (P4).* Participants P1, P2, P3, P5, P6, and P8 all cited frequent use of the Jira issue tracking software on their teams to track assigned work and communicate progress.

Updating an issue ticket when a task is completed is not only to help individuals stay organized and on track, but for other team members' awareness as well. Everyone on the project can view all the tasks and their statuses, so the tracking system, *"makes for more structured and connected work. [The engineers] don't lose information along the way and keep in sync with each other" (P4).* Redundancies are mitigated with labels indicating which team members are working on which tasks. Participants can be added on to an existing issue and make comments or changes, if needed. If a conversation with another team member took place that assisted in the completion of a task, that can be indicated within an issue as well. In the event another similar dilemma arises, an engineer can immediately reference the details, which point to exactly where to find the archived notes, and save themselves from

repeating the conversation. The platform can also address the challenge of continuity documentation by offering a standardized, yet customizable, method of showing how a conclusion was reached through a timeline of tasks, subtasks, and details within each ticket.

4.5.2 Shared Workspace

RE teams almost always work in close physical proximity. Therefore, the collaborative tools employed often serve the primary purposes of documentation and reference, rather than pure communication. This is most evident with the use of a shared workspace, commonly referred to as a "wiki." Five of the eight participants discussed using the Confluence web-based wiki system to aggregate group knowledge *(P1, P3, P4, P5, P8)*. All of the resources and background information on a project are uploaded to the wiki for the team to reference at any time. How a team maintains the wiki as the project progresses is unique to each team's preferences. One possible use case is creating a running tasklist that mirrors the one in the issue tracking software. Each task can be expanded into its own page where files can be uploaded, accessible to everyone on the team.

While the issue tracking software is primarily managing workflow, the wiki houses relevant artifacts for each task, such as screen captures and detailed notes taken by the engineer assigned to the task. This shared workspace serves as a single, central location for any information that may need to be referenced later for any purpose, including writing the final report. For instance, if the team lead suggests adding evidence in the report to support a conclusion, engineers can often *"copy and paste technical details straight from their Confluence pages" (P4)*. Alternatively, if an engineer knows that a problem they currently face has been solved in the past by themselves or a colleague, they can navigate straight to the wiki page where

the solution was documented, rather than combing through stacks of old notes or bothering the colleague.

In the issue tracking software, an engineer can simply indicate that a conversation with a colleague occurred. The shared workspace would be where one could elaborate on the details and outcomes of the discussion, available to be referenced at a later time if necessary. Continuity in the event of a personnel change also becomes more convenient through use of the shared workspace. If well-maintained, the outgoing engineer's pages document precisely how they arrived at the current state and everything they found along the way. The incoming engineer would not need to recreate all the analysis, based off of a few verbal pointers, in order to arrive at the same level of comprehension. Furthermore, since the pages can be viewed by everyone, redundancies that slipped past the issue tracking system can be identified in the shared workspace.

4.5.3 Team Chat Platform

Though verbal communication is indispensable in collaborative RE, some occasions call for a digital means of transmitting a message to the team. Through a group chat service, an engineer is able to get a message out to the greatest number of people in the shortest amount of time. This method allows the recipients flexibility in their response times and opportunity to alter responses, unlike if one were to yell across the room and expect an immediate, definitive response. *"Some of [the engineers] don't like to be disturbed when they're really in the zone, so being able to have a message put aside to get back to later allows them to stay aware of their surroundings while maintaining focus" (P3).*

As discussed previously, when a short conversation takes place, the engineer receiving assistance bears the responsibility of documenting the takeaways in the task's

wiki page. This process happens automatically in a team chat platform, where records of conversations are maintained by the software and easily retrievable down the road. Chat services are also customizable, allowing integration with issue tracking software to send automated message updates to the team on the status of issue tickets. This capability keeps everyone aware of the team's progress and ongoing endeavors. Additionally, upon encountering something potentially worth looking into, an engineer could either search the chat history or poll their colleagues to find out if it has already been investigated. A positive response could save them from falling victim to redundant work.

4.5.4 Ghidra Server

On top of sharing updated notes and documentation of work, keeping all project files synchronized and up to date for the whole team can be equally important to productivity. When major changes and breakthroughs occur, the whole team should not only be made aware but have access to the most current version of the files. The Ghidra Server capability allows engineers to push and pull changes to the team's repository of project files, such as comments, function name changes, or addition of files. Integrated into a popular disassembling tool, the service is *"no fuss and gets the job we need it to do done very well. It's not very resource intensive and it's nice to not really feel the presence of it" (P4).* Being able to reference past changes can also be helpful alongside documentation for continuity purposes in a handoff. For the individual, the server operates as a backup in the event of a system crash and mass data loss. Conflicts in commits made to the repository may arise, alerting the engineers to redundancies occurring in the team's work. Rollbacks to previous versions are occasionally necessary as a result *(P4).*

While not all interviewed engineers use Ghidra as their disassembler of choice, the

shared version control functionality has enticed many to either make the switch or use it in conjunction with their preferred disassembler for the value it provides. For instance, participants P5 and P6 recall switching to Ghidra only to share project files with collaborators, while continuing to use IDA Pro for disassembly. Participants P1, P2, P4, and P8 use Ghidra for their day-to-day work and find the shared project functionality well-integrated and easy to use.

4.5.5 Workflow Integration

In a teamwork environment, *"efficiency is easily the most obvious thing to look to improve" (P8).* Increased efficiency is attained with better communication, resulting in *"being able to get through more system and programs in less time, but also doing so in a more thorough manner."* When executed well, the client also receives a higher level of assurance that the team has found everything there is to be found.

Reverse engineers of all experience levels have recognized the value that collaborative tools such as issue tracking software, shared workspaces, and chat services bring to their team's efficiency and productivity. Despite that widespread acknowledgement, full integration of the tools into their existing workflow has proven difficult. *"A lot of the time, it's about breaking old habits and enforcing discipline [of using the tools] for the good of the team" (P5).* Though notes are still taken individually, team leads are encouraging a new mindset of *"collaborative notetaking,"* where collective knowledge and awareness is a constant consideration.

4.6 Limitations and Threats to Validity

A limiting factor of the research was the sample size of eight interview participants. A larger sample size may have revealed opposing opinions on the needs and priorities of engineers in a collaborative environment. Discussions were also limited

by confidentiality concerns with specific details of the participants' workflows. The researcher mitigated this limitation by focusing on the interactions between engineers and use of collaborative tools, and generalizing workflow processes.

To avoid asking leading questions influenced by previous interviews, the researcher used a standard survey instrument in each interview. Acquiescence bias, where participants tend to just agree with the interviewer, was mitigated with open-ended questions that appealed to their real-life experiences. Additionally, participants may have been hesitant to be candid about their past trials and failures if they deemed it hurtful to their pride or professional reputation. To mitigate this effect and any other retrospective bias, the researcher did not imply or refer to task completions as accomplishments, instead redirecting attention to the collaborative elements of interactions between the engineers involved in the process. The researcher also did not demand technical details where no value would be added to narrative regarding collaboration.

The process of data analysis involved drawing conclusions from interview data. To mitigate making presumptions with under-developed evidence, the researcher ensured multiple sources could support the findings presented in the results and analysis.

All of the research's participants currently work for the Department of Defense (DoD) in some capacity. Though some drew on their experiences in the private sector, the study's results' generalizability is limited to reverse engineering work in the DoD.

4.7 Chapter Summary

This chapter detailed the collaborative RE workflow as recalled by the industry experts interviewed. The process spanned from the initial triage of the software system to the final report written presenting the teams findings. Participants identified the needs of an RE team and challenges encountered on a regular basis, and discussed how the tools they commonly employ meet those needs. The chapter answers

the research question: *How is collaboration currently conducted in reverse engineering?* The next chapter will summarize the contributions of this research and make recommendations for future work.

V. Conclusions

Software reverse engineering (RE) in a team can easily prove less efficient than individual RE without robust collaboration practices in place. The collaborative RE process can differ from one project to the next, presenting new roadblocks that require creative solutions to overcome. This research consulted expert reverse engineers to identify their needs throughout the collaborative workflow and explore how specific tools work to fit those needs. The interviews drew on the participants' experiences that involved the use of collaborative tools and practices. The results successfully mapped all of the participants' needs to one or more tools employed by their teams, and explained each connection. This chapter summarizes the research conclusions and opportunities for future research.

5.1 Research Conclusions

The survey instrument was designed with the intent of answering one core research question by addressing four supporting questions. The first supporting question, "What does the collaborative RE workflow look like, from start to finish?" was answered by identifying common decision points in the CDM portion of the interviews. The result was the complete collaborative RE workflow presented. Establishing this workflow is essential to understanding where their needs and challenges occur in the larger timeline of events.

Through the CDM's deep dive into each decision point and factors that affected each decision outcome, the interviews addressed question 2, "What are the user needs and challenges unique to the collaborative process?" Though each participant used unique scenarios to convey their needs, several common themes emerged across the interviews. Maintaining concurrent documentation, mitigating redundancies, and man-

aging continuity documentation while collaborating were a few of the major challenges that the experts encountered.

Question 3 was posed to the participants directly at the start of the interview, "What collaborative tools do RE teams currently use?" Issue tracking software, shared workspaces, team chat platforms, and shared file version control were the tools identified as able to fulfill their collaborative needs. The participants also explained how those collaboration tools affected each course of action taken throughout the CDM timeline, addressing question 4, "How are those collaborative tools integrated into a team's workflow?" Optimizing integration remains an ongoing challenge as reverse engineers face difficulties with breaking old habits to maximize the tools' values.

The conclusions drawn from each of the four supporting questions contribute to answering the main research question, "How is collaboration currently conducted in reverse engineering?" The results show the complete workflow of an RE team and explain how the collaborative tools discussed were found to meet one or more of the team's needs.

5.2 Recommendations for Future Work

Given the results of this research, there are several opportunities for future work. One possibility involves consolidating the various collaborative tools currently used. Experts seem satisfied with the functionalities the tools provide and their ability to fulfill the team's needs. For instance, one major factor in the appeal of the Ghidra server functionality lies in its seamless integration with a widely used RE analysis tool. Meanwhile, other tools discussed, such as issue tracking and chat software, lack the same convenience. Each tool stands alone and requires additional effort to capitalize on their value. A common complaint when introducing new tools to an existing workflow is *"unnatural and cumbersome" (P2)* implementation. Combining

the unique features of all the commonly used tools into a single solution could afford users greater improvements to the collaborative RE experience with less overhead.

Currently, the Ghidra Server capability allows collaborators to maintain a complete log of commits and past versions of a shared project. The viewable details include timestamps, the user who made the commit, and comments. Another possible avenue for future work involves including provenance data with each commit in Ghidra Server. Provenance data shows the actions that the engineer took within the disassembler in chronological order. This information could assist one collaborator in understanding how another was able to solve a problem or reach a conclusion within a commit. As discussed previously, a provenance plug-in was built for Binary Ninja. However, Ghidra Server is a popular, well-established tool that could also benefit from additional visualization features for efficient collaboration.

This research has established that in-person communication plays a large role in an RE team's ability to solve problems quickly. Shoulder-surfing, posing questions to the entire room, and impromptu brainstorming sessions are common practices that are difficult to replicate to the same effect without the team members' physical presence. As many other software-related fields are beginning to conduct operations remotely with certain amounts of success in the new decade, reverse engineers remain adamant that it isn't plausible and *"[working remotely] just doesn't work" (P4).* Future research could investigate the specific aspects of RE that make the work unsuitable for remote collaboration.

5.3 Closing Remarks

Working in teams seems like an obvious solution to reverse engineering software systems better and faster. However, collaboration does not equate to efficiency. Efforts must be made to identify emerging challenges and their solutions. This research

investigated the workflow of RE teams, the needs that they encounter, and the collaborative tools employed as solutions for those needs. The results indicate that collaborative tools are capable of organizing team logistics and managing communication channels, so that engineers can focus on analyzing the code and developing new exploits. Establishing an understanding of how collaborative tools meet RE teams' needs reveals opportunities for tool developers to enhance processes and optimize productivity in the future.

Appendix A. Request for Human Experimentation

This appendix provides the request for exemption determination from human experimentation requirements, submitted on 17 September 2021.

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (AETC)

17 September 2021

MEMORANDUM FOR AFIT HUMAN RESEARCH PROTECTION PROGRAM (HRPP), OFFICE OF SPONSORED PROGRAMS AND RESEARCH (ENR)

FROM: AFIT/ENG

SUBJECT: Principal Investigator Cover Letter for Exempt Determination Request for Examining Collaborative Cognition in Software Reverse Engineering

1. Request AFIT HRPP review and approval of the Exempt Determination Request protocol named above which should be considered as a freestanding protocol.

2. As principal investigator (PI), the undersigned affirms that the protocol complies with the requirements for exempt research set forth in Federal code and the DoD, Air Force, and AFIT instructions implementing it. In addition, the undersigned agrees to:

   a. Ensure that all exempt research conducted under this protocol will conform to the written, approved document, including any restrictions imposed during the approval process. The funding and resources for this research have been procured/acquired to conduct this project as submitted in the protocol. The funding source is:

Funding Agency/Organization: AFIT/ENR	
Funding Amount: $13,800	JON#: 21-901

   b. Personally conduct and supervise the study and be responsible for the conduct of all persons acting on behalf of the Principal Investigator.

   c. Monitor the progress of this research and notify the AFIT HRPP in writing within 24 hours of any unexpected event, unanticipated problem, safety concern or medical misadventure.

   d. Promptly notify AFIT HRPP, if either the risk or the benefit of the research appears substantially different from those represented in the protocol, or if early results clearly resolve the hypothesis.

   e. Ensure all individuals assisting in the study are adequately trained, and aware of their responsibilities.

   f. Maintain and retain all study and protocol documents as required by the protocol and DoD regulations.

   g. Conduct this research in compliance with the principles of Human Subjects Research found in the Belmont Report: 1. Respect for Persons requires that subjects, to the extent they are capable, be

given the opportunity to choose what will or will not happen to them. The informed consent process contains three elements: information, comprehension and voluntariness. 2. Beneficence closely relates to the risk/benefit assessment which is concerned with the probabilities and degree of possible harm and anticipated benefits. 3. Justice addresses moral requirements that there be fair procedures and outcomes in the selection of research subjects. Individual justice ensures that the selection of subjects is done in fairness. Social justice requires that distinction be drawn between who should and who shouldn't participate in any particular kind of research based on the ability of individuals to bear burdens and on the appropriateness of placing further burdens on already burdened persons.

3. As the Principal Investigator of this research study I assume responsibility for the overall management of this protocol and ensuring each investigator meets the reporting requirements of the attached Conflict of Interest Disclosure Checklist. I agree to notify AFIT HRPP in writing if any conflict of interest within the research team exists or arises during the project.

4. In accordance with DoD 8520.02, only Principal Investigators with a CAC card may provide an electronic signature as permitted on this template. For Principal Investigators who do not have a CAC card, please print the completed application, provide a handwritten signature, and scan the document so that it may be attached to an email for submission.

WAYNE HENRY/LT COL/ASST PROFESSOR
Principal investigator

Appendix B. Ethics Approval

This appendix provides the approval for the exemption request for human experimentation requirements protocol number REN2021013R from the Air Force Institute of Technology. The study was approved on 27 October 2021.

DEPARTMENT OF THE AIR FORCE
AIR EDUCATION AND TRAINING COMMAND
AIR FORCE INSTITUTE OF TECHNOLOGY
WPAFB, OH

MEMORANDOM FOR AFIT/ENG
ATTN: LT. COL. WAYNE C. HENRY

FROM AFIT/ENS
2560 Hobson Way
WOAFB, OH 45433

SUBJECT: Exempt Determination Official (EDO) Review of REN2021013R, "Examining Collaborative Cognition in Software Reverse Engineering"

References: (a) 32 CFR 219, *Protection of Human Subjects*
(b) DoDI3216.02_AFI40-402, *Protection of Human Subjects and Adherence to Ethical Standards in Air Force Supported Research*

In accordance with Reference (b), Enclosure 2, Paragraph 10(d), an EDO review has been completed for the above-referenced activity. It was determined that this activity qualifies as research involving human subjects and is exempt under Section 219.104(d)(2)(i)/(ii) of Reference (a) based on the following rationale:

This study will evaluate the practices of software reverse engineers in DoD while collaborating synchronously and asynchronously. No PII will be collected. Risk in similar to that experienced in an every-day office environment. The questions should not put the subjects at risk personally, financially, or politically. The results of the interview will be used to conduct a systematic investigation designed to understand software reverse engineers' needs and goals and examine challenges and recommendations for future research and development.

Prior to implementation, if there are any questions or changes to this activity that may alter the findings of this exempt determination, please contact Dr. Seong-Jong Joo at seong-jong.joo@afit.edu.

Seong-Jong Joo, PhD.
Professor of Logistics & Supply Chain Management
AFIT/ENS

Appendix C. Survey Instrument

Survey Instrument

The purpose of the study is to examine the collaborative software reverse engineering (RE) process. The study will examine the processes and tools employed today by reverse engineers in the DoD and identify common challenges and recommendations for future research and development. The data collected today will help in forming a cognitive model of those regularly involved in such processes to better understand their needs.

The interview session will take approximately 2 hours, not to exceed 3 hours. You may request a 10-minute break at any time.

Preliminary:

1. Briefly describe your educational background. To what extent was RE a part of your formal education?
2. How many years of professional experience RE do you have thus far?
3. Please describe/summarize your professional experience. What kind of RE/How much collaboration?

Part 1:

1. Without going into to the team RE process just yet, describe the tools/techniques utilized today while performing software RE with others, specific to the collaborative environment. If they have evolved over the course of your career, please specify how.
2. What are the limitations of the tools/techniques used today for collaboration in RE?

Part 2:

1. How does the collaborative RE workflow look for the team? Recall a specific, past incident in your career, where you have had to collaborate with other reverse engineers to complete a RE project/task.
   (a) Take us through a big-picture timeline of what happened first.
   (b) (For each CDM decision point,) what was the major challenge, and how was it addressed?
      (i) How did the involved engineers' level of experience (at the time) contribute to the handling of the situation?
      (ii) How did the integration of collaborative tools affect the courses of action taken?
      (iii) From this decision point, what were some lessons learned? Did any of these lessons reappear later in the project?
2. (Reiterate previous lessons learned,) did any of these lessons have an impact on a future collaborative RE processes?
3. Based on this experience, what improvements to the collaborative RE tools/techniques could have been of further assistance?
   (a) Possible risks or tradeoffs to these improvements?
   (b) What outcomes would indicate success/How specifically would it improve the collaborative experience?

Closing Comments:

Bibliography

1. Executive order no. 14028. 86 FR 26633, 2021.

2. E.J. Chikofsky and J.H. Cross. Reverse engineering and design recovery: a taxonomy. *IEEE Software*, 7(1):13–17, 1990.

3. Christopher Eagle and Timothy Vidas. Next generation collaborative reversing with ida pro and collabreate. *Black Hat Briefings,(Las Vegas, USA)*, 2008.

4. Christoph Treude, Fernando Figueira Filho, Margaret-Anne Storey, and Martin Salois. An exploratory study of software reverse engineering in a security context. In *2011 18th Working Conference on Reverse Engineering*, pages 184–188. IEEE, 2011.

5. Daniel A. Quist and Lorie M. Liebrock. Visualizing compiled executables for malware analysis. *6th International Workshop on Visualization for Cyber Security 2009, VizSec 2009 - Proceedings*, pages 27–32, 2009.

6. Daniel Votipka, Rock Stevens, Elissa Redmiles, Jeremy Hu, and Michelle Mazurek. Hackers vs. testers: A comparison of software vulnerability discovery processes. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 374–391, 2018.

7. Daniel Votipka, Seth Rabin, Kristopher Micinski, Jeffrey S. Foster, and Michelle L. Mazurek. An observational investigation of reverse engineers' process and mental models. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, page 1–6, New York, NY, USA, 2019. Association for Computing Machinery.

8. Wayne C. Henry and Gilbert L. Peterson. Exploring provenance needs in software reverse engineering. *Proceedings - 2020 13th Systematic Approaches to Digital Forensic Engineering, SADFE 2020*, pages 57–65, 2020.

9. Markus Wagner, Alexander Rind, Niklas Thür, and Wolfgang Aigner. A knowledge-assisted visual malware analysis system: Design, validation, and reflection of kamas. *Computers and Security*, 67:1–15, 2017.

10. Mariano Ceccato, Paolo Tonella, Cataldo Basile, Bart Coppens, Bjorn De Sutter, Paolo Falcarin, and Marco Torchiano. How professional hackers understand protected code while performing attack tasks. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pages 154–164. IEEE, 2017.

11. C. Cifuentes. An environment for the reverse engineering of executable programs. In *Proceedings 1995 Asia Pacific Software Engineering Conference*, pages 410–419, 1995.

12. S Megira, A R Pangesti, and F W Wibowo. Malware analysis and detection using reverse engineering technique. *Journal of Physics: Conference Series*, 1140:012042, dec 2018.

13. Zhibo Liu and Shuai Wang. How far we have come: Testing decompilation correctness of c decompilers. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2020, page 475–487, New York, NY, USA, 2020. Association for Computing Machinery.

14. Ghidra. `https://www.nsa.gov/resources/everyone/ghidra/`. Accessed: 2022-01-17.

15. Ida pro. `https://hex-rays.com/ida-pro/`. Accessed: 2022-01-17.

16. Ghidra vs ida pro. https://allabouttesting.org/ghidra-vs-ida-pro-which-one-is-better/. Accessed: 2022-01-17.

17. Binaryninja. https://binary.ninja/. Accessed: 2022-01-17.

18. Byte.how. Collaborative reverse engineering with ghidra server, Sep 2020.

19. Collare v0.3. https://pythonrepo.com/repo/Martyx00-CollaRE. Accessed: 2022-01-17.

20. Idarling. https://github.com/fidgetingbits/IDArling. Accessed: 2022-01-17.

21. Wayne C. Henry and Gilbert L. Peterson. Sensorre: Provenance support for software reverse engineers. *Computers and Security*, 95, 2020.

22. Chris North, Remco Chang, Alex Endert, Wenwen Dou, Richard May, Bill Pike, and Glenn Fink. Analytic provenance: Process + interaction + insight. page 33, 2011.

23. David Christian and Shmuel Rotenstreich. An evaluation framework for distributed collaboration tools. In *2010 Seventh International Conference on Information Technology: New Generations*, pages 512–517, 2010.

24. Filippo Lanubile, Christof Ebert, Rafael Prikladnicki, and Aurora Vizcaíno. Collaboration tools for global software engineering. *IEEE Software*, 27(2):52–55, 2010.

25. Diomidis Spinellis. Git. *IEEE Software*, 29(3):100–101, 2012.

26. About - git. https://git-scm.com/about/branching-and-merging. Accessed: 2022-01-17.

27. About - git. https://git-scm.com/about/distributed. Accessed: 2022-01-17.

28. Visual studio live share. https://visualstudio.microsoft.com/services/live-share/. Accessed: 2022-01-17.

29. Teletype for atom. https://stackshare.io/teletype-for-atom. Accessed: 2022-01-17.

30. The heilmeier catechism. https://www.darpa.mil/work-with-us/heilmeier-catechism. Accessed: 2021-12-06.

31. G.A. Klein, R. Calderwood, and D. MacGregor. Critical decision method for eliciting knowledge. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(3):462–472, 1989.

32. Hana Harenčárová. Structured analysis of critical decision method data – emergency medicine case study. *Human Affairs*, 25:443–459, 10 2015.

33. Atul A Gawande, Michael J Zinner, David M Studdert, and Troyen A Brennan. Analysis of errors reported by surgeons at three teaching hospitals. *Surgery*, 133(6):614–621, 2003.

34. Emily S. Patterson, Richard I. Cook, and Marta L. Render. Improving Patient Safety by Identifying Side Effects from Introducing Bar Coding in Medication Administration. *Journal of the American Medical Informatics Association*, 9(5):540–553, 09 2002.

35. Michael Quinn Patton. *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications, 2014.

36. Larry E. Wood. Semi-structured interviewing for user-centered design. *Interactions*, 4(2):48–61, March 1997.

37. Beth Crandall, Gary A. Klein, and Robert R. Hoffman. *Working Minds: A Practitioner's Guide to Cognitive Task Analysis*. The MIT Press, 07 2006.

38. Larry E. Wood and John M. Ford. Structuring interviews with experts during knowledge elicitation. *International Journal of Intelligent Systems*, 8(1):71–90, 1993.

39. Maxqda: Qualitative data analysis software. 2021. Available: https://www.maxqda.com.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–03–2022	Master's Thesis	Sept 2021 — Mar 2022

4. TITLE AND SUBTITLE

Investigating Collaboration in Software Reverse Engineering

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

2d Lt Allison M. Wong

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-22-M-075

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Reverse engineering (RE) is a rigorous process of exploration and analysis to support software design recovery and exploit development. The process is often conducted in teams to divide the workload and take full advantage of engineers' individual expertise and strengths. Collaboration in RE requires versatile and reliable tools that can match the environment's unpredictable and fluid nature. While studies on collaborative software development have indicated common best practices and implementations, similar standards have not been explored in reverse engineering. This research conducts semi-structured interviews with reverse engineering experts to understand their needs and solutions while working in a team. The results describe an array of major challenges that are each addressed by employing tools such as issue tracking software, shared workspaces, and version control systems. Such tools support documentation and continuity, while mitigating redundancies in concurrent work. Though the value of these tools are acknowledged by the experts, seamless workflow integration remains a challenge. The identification of current needs and practices offers additional opportunities for collaborative tool developers to aid reverse engineers.

15. SUBJECT TERMS

collaboration, documentation, reverse engineering, workflow

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt Col Wayne C. Henry, AFIT/ENG
U	U	U	UU	72	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, ext 7243; wayne.henry@afit.edu