Exploring Provenance Needs in Software Reverse Engineering

Wayne C. Henry, Gilbert L. Peterson
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, USA
wayne.henry@afit.edu, gilbert.peterson@afit.edu

***Abstract*—Reverse engineers are in high demand in digital forensics for their ability to investigate malicious cyberspace threats. This group faces unique challenges due to the security-intensive environment, such as working in isolated networks, a limited ability to share files with others, immense time pressure, and a lack of cognitive support tools supporting the iterative exploration of binary executables. This paper presents an exploratory study that interviewed experienced reverse engineers' work processes, tools, challenges, and visualization needs. The findings demonstrate that engineers have difficulties managing hypotheses, organizing results, and reporting findings during their analysis. By considering the provenance support techniques of existing research in other domains, this study contributes new insights about the needs and opportunities for reverse engineering provenance tools.**

***Keywords*-Reverse Engineering; Qualitative User Study; Visual Analytics-Provenance; Interviews**

I. Introduction

Malware is a common tool in digital theft, corporate and national espionage, spam distribution, and attacks on infrastructure [1]. Government institutions and businesses rely heavily on software reverse engineers to take malicious software apart and thereby better understand it and collect relevant evidence [2]. However, as cybersecurity threats escalate and increase in complexity, it has become increasingly difficult for software reverse engineers to keep pace with the demand [3].

Unfortunately, analyzing software binaries is a time-consuming and complex process [4]. While advancements in data science and automation assist certain aspects of the analysis pipeline, the complexity and uncertainty encountered for real decision-making makes human involvement significant and inevitable [4]. Without the benefit of source code and documentation, reverse engineers have to examine low-level data representations and mentally reconstruct the functionality of a program. Visualization techniques are often combined with human-computer interaction methods to support the users' cognitive capabilities during data exploration. Multiple studies have attempted to understand the problems of reverse engineers and developed innovative solutions to address their needs [5]–[7]. Yet, large gaps still exist for reverse engineers managing findings, collaborating, and communicating findings with stakeholders [2], [8].

In other highly-exploratory scientific fields (e.g., medical analytics [9] and digital forensics [10]), *analytic provenance* techniques have been explored as a potential solution to these problems. Analytic provenance systems capture both the interactive data exploration process and the accompanied reasoning process during sensemaking relieving the user from the burden of managing their discoveries [11]. The provenance data is frequently visualized to provide different types of support to users, such as recall, replication, action recovery (undo, redo), collaborative communication, and presentation of findings [12]. However, it is also largely unclear how the success of systems in one domain can be applied to an unrelated domain. A study of the work processes and challenges is needed to address reverse engineering provenance needs.

To the best of our knowledge, no studies have investigated the processes and challenges of reverse engineers for the development of analytic provenance tool needs. This study presents findings from interviews with expert reverse engineers reporting the domain-related difficulties managing hypotheses, organizing results, and reporting findings. An analysis of the results indicate that analytic provenance tools may serve as an effective cognitive aid, allowing reverse engineers to recall, collaborate, and present formative details of their strategies. In addition to describing the participants' practical work processes and challenges, the study also considers how these challenges could be addressed through provenance support techniques.

II. Related Work

Relatively few studies have examined the workflow and processes employed by reverse engineers, which has made it difficult for industry to develop software that specifically targets reverse engineers. Treude, et al. explored the work processes of software reverse engineers in a security context [2]. Baldwin, et al. further identified the limitations of visualizations within the reverse engineering domain and developed a methodology to identify associated requirements from two specialized groups of assembly language developers [13]. Through surveys, observations, and interviews with these groups, the researchers identified a range of tooling needs. Kienle and Muller have comprehensive discussions on reverse engineering tool development in terms of require-

ments, software architectures, and tool evaluation criteria [8], [14]. While these works provide the groundwork for this study, they do not investigate the analytic provenance needs of software reverse engineers.

Meanwhile, in the related hacker community, several empirical studies have explored the tool needs of users, including during reverse engineering tasks [15], [16]. Many of the tools described in such studies were created in an ad-hoc manner and are well-known for being difficult to use [4]. To cope with these difficulties, hackers require high levels of tolerance, patience, and perseverance. While the hackers in these previous studies performed many other activities beyond reverse engineering (e.g., network penetration testing), the progressive methodology employed in these studies for examining the discovery process, methods, and visualization needs of the hackers influenced the design of this study.

Several studies have also captured and characterized the work practices and analytical processes of individual or collaborative analysis through a qualitative approach. For example, Pirolli and Card studied intelligence analysts and developed a notional model of the analytic processes they follow [17]. In addition, Chin, et al. conducted an observational case study with professional intelligence analysts in which participants worked on real-world scenarios, either as individual analysts or as an investigative team [18]. The researchers identified various characteristics of the analytical processes of intelligence analysts, such as the investigative methodologies they apply, how they collect and triage information, and how they identify patterns and trends. Understanding the analytical processes of reverse engineers, including how they are currently capturing and using provenance, will be helpful for identifying gaps in their provenance support technologies.

III. Method

This section details the exploratory qualitative methodology employed in this study. The researchers conducted semi-structured interviews with five experienced engineers. Research by Creswell [19] and Wood [20] influenced the study design. The goal of the interview process was to understand the work practices of reverse engineers through exploring their processes, tools, challenges, and visualization needs.

A. Participants

The target population was professional software reverse engineers. Individuals with fewer than two years of hands-on experience in reverse engineering were excluded from the sample.

Table I summarizes the demographics of the participants. On average, the participants had 8.6 ($\pm$ 4.6) years of reverse engineering experience in the security sector. Their duty titles included "malware analyst" *(P1, P2)*, "cyber tools analyst" *(P3, P4)*, and "software analyst" *(P5)*. Two participants held PhDs in computer engineering, one held two Master's degrees (computer science, MBA), and two held Bachelor's degrees (mathematics and computer engineering).

Table I: Participants' backgrounds.

Participant	Duty Title	Exp. (Yrs.)	Education
P1	Malware Analyst	4	MS CS
P2	Malware Analyst	13	BS Math
P3	Cyber Tools Analyst	10	PhD CE
P4	Cyber Tools Analyst	5	BS CE
P5	Software Analyst	11	PhD CE

Gaining access to participants who already possessed a strong understanding of software reverse engineering was a common limitation in previous studies [4], [5], [21]. The participants in this study self-reported their professional experience. To recruit experienced participants, the researchers emailed contacts at national security organizations through personal and professional networks. These contacts forwarded the request to engineers within their organizations. The participants were under no internal or external pressure to participate.

B. Procedure

The in-person semi-structured interviews each began with a brief introduction describing the purpose of the interview. Each interview lasted approximately two hours. The interviews were performed off-site due to security limitations but near the participants' work locations.

The semi-structured interview questions were pilot-tested and refined based on feedback from a reverse engineering expert with 12 years of experience. The questions were designed to gather information on task analysis workflows, tools, challenges, and visualization needs. While the study aimed to study the role and uses of provenance, the researchers did not introduce the term *provenance* in the interview questions to avoid leading questions. The questions were meant to kindle the discussion and off-topic elaborations were permitted during the interview. Each interview discussed the following questions:

1) What work tasks do you perform during reverse engineering?
2) What key decisions do you make during the analysis?
3) What tools do you regularly use and how do you use them?
4) What are the results of analysis and how are they shared with external stakeholders?
5) How are findings shared between team members?
6) How do time constraints modify your analysis process?

With the permission of the participants, the interviews were audio recorded and extensive notes were taken. The researchers encouraged participants to speak freely about the issues encountered during the binary analysis process. Prompts were also used to encourage participants to provide more detail. Given the participant-driven nature of

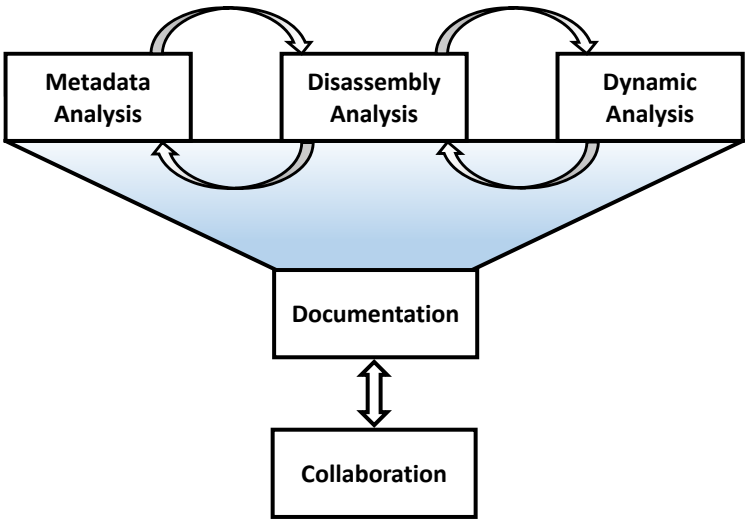

Figure 1: Overview of the reverse engineering workflow process.

exploratory interviews, there were varying levels of responses. For example, one participant talked at length about teamwork concerns, while another focused primarily on workflow challenges.

Due to security concerns related to the practical application of the techniques discussed, participants were encouraged to provide generic examples and practices to avoid disclosing national security information.

C. Data Analysis

Following the interviews, the audio data was transcribed and compared with the researchers' notes for accuracy and completeness. The researchers followed an open coding strategy for labeling and categorizing data using the data analysis software MaxQDA [22]. Common practices, tools, challenges, and constraints were grouped into high-level themes, which were refined as more data was gathered. The comments were organized based on the associated tasks and problems within each theme. A draft version of the analysis document was distributed to two experts involved in the study to verify the findings.

IV. Findings

This exploratory study sought to understand the participants' reverse engineering processes in order to ultimately identify opportunities for tool development. New reverse engineering tools will likely always be needed to address the rapidly changing software landscape. By analyzing the interview data, the researchers identified several potential tool attributes that could enhance reverse engineers' investigative and analytic capabilities.

This section presents the findings from the semi-structured interviews. Each participant described challenges explaining their typical workflow due to the broad range of reversing projects encountered. However, several overlapping areas emerged. Based on the interview data, the participants identified five common processes in the reverse engineering workflow: *metadata analysis, disassembly analysis, dynamic analysis, documentation,* and *collaboration*. The evolution between these themes is captured in Fig. 1.

To our surprise, we generally observed the same processes and methods were used by the reverse engineers performing malware analysis as those that were seeking vulnerability discovery or general program understanding. However, there were differences in what the participants prioritized based on their role (e.g. malware analysts focused initially on network calls while others looked at memory management functions). However, because our focus is on the high-level processes and methods used, we discuss both groups together in the following sections.

A. Metadata Analysis

Metadata analysis was the first procedural step described by all participants. The term *metadata* in this context refers to static analysis data that provides information about the binary prior to deeper inspection. Engineers perform metadata analysis to form initial impressions of files with little or no prior knowledge, as stated by one subject, *"script as much as we can to point the analyst in the right direction" (P3)*. Sources of metadata include the file name, size, header information, file structure, and compiler artifacts. These data sources assist in generating a broad understanding of the compiled program and potentially in forming hypotheses for further examination.

The specific methods and tools used vary according to the reverser's objectives, the file under analysis, and the extent of information already known about the file. *"[We use] a lot of utilities for understanding the header information for*

files; sometimes we have files that we're not really sure what they are" (P2). These utilities include hex editors, binary difference tools, Linux utilities (e.g., file, objdump), packer detection (e.g., PeID), the Windows Sysinternals suite, PE viewers (e.g., CFF Explorer, PE Explorer), and many others depending on the situation. The metadata analysis phase is the quickest of the five phases described, *"usually completed within an hour" (P2)*.

Given the broad range of tools required, the participants described their reluctance with non-scriptable stand-alone tools. Instead, the participants preferred tools that could be integrated into other tools. *"There are too many tools we use all the time that we build scripts to streamline our processes. Ideally, new capabilities could tie into one of the [tools] we already use" (P3)*. An analytic provenance tool could be non-intrusively integrated into existing software analysis tools, capturing and then displaying the visualized history data in order to minimize the impact on the reverse engineers' workflow.

B. Disassembly Analysis

Each participant reported that the majority of their time is spent statically analyzing code in a disassembler. All participants stated that IDA Pro was their disassembler of choice with Binary Ninja and Ghidra as secondary options. When deciding which disassembler to use, all participants indicated that they would often examine a given binary executable with multiple disassemblers at the same time in order to piece together their understanding. *"Different tools produce different representations which can be helpful in piecing together the functionality of the binary" (P5)*.

In describing the strengths and weaknesses of the disassemblers, the participants highlighted the power of plugins in supplementing the base tool. Participants *P1*, *P2*, and *P4* stated that they use plugins "heavily" during analysis, while P5 reported using plug-ins only "occasionally." Due to classification concerns, the participants did not provide details on the specific plug-in capabilities. However, they described desirable attributes of the tool's plug-in interface, including ease of development, API documentation, and an active development community. These factors contributed to the choice of IDA Pro as the preferred tool in this phase of analysis, followed by Binary Ninja.

The participants next described their analytic process and workflow. Many malware binaries are stored in obfuscated form and only deobfuscated at execution time to complicate the reverse engineering process. This obfuscation is commonly referred to as *packing*. Once the binary is unpacked the participants perform the same overview analysis described below. At this stage, reverse engineers examine high-level details such as system calls, strings, user interface elements, or other structural information. The comprehension of these details is further refined through subsequent goal-driven exploration activities.

> *[We are] looking for a handful of things that might indicate what kind of inter-process communication this binary has, either internally or externally. Statically, we want to find out what [the program] interconnects with; how it interacts with the file system, network, or something embedded inside it (P2)*.

Analyzing program interaction through system calls is a critical step in the engineers' workflow. *"API calls or custom programming calls tell us a lot of information about what [the binary] is doing" (P1)*. Reverse engineers utilize the import table and an iterative process of examining the system calls to piece together how the program functions.

Based on findings during the overview, reverse engineers next shift their attention to program sub-components and behaviors, including function call mapping, variable use, and program control flow. For example, while investigating a piece of malware, P2 recalled seeing `URLDownloadToFile` was called in a function call map. This piqued his interest because "it's frequently used to download malware from a remote server or website." He could then infer program behavior based based on past experience, "usually you see internet checking activity if the malicious software is trying to download additional malware."

C. Dynamic Analysis

Participants indicated that dynamic analysis is as an optional, but useful analysis step during focused experimentation. Dynamic analysis traces the application's events at run-time, allowing the engineer to inspect stack variables and develop an understanding of the data flow. The participants frequently used debuggers, when possible, to validate their hypotheses and iterating their findings back into the disassembly analysis. *"Dynamic analysis [is] more useful once you already have an understanding of what's happening because then you can look in the stack, you can look at a stack trace and figure out what's there, or stop it at a certain point" (P3)*.

The most frequently used dynamic analysis tool by the participants was the debugger (e.g., IDA Pro's integrated debuggers, OllyDbg, or WinDbg), followed by virtual machine emulation tools (e.g., PANDA, Intel PIN), and occasionally fuzzers (e.g., Peech, AFL, and Sully). Participant *P5* described using fuzzers primarily for long-term, exhaustive analyses, which were not common-place in his work center.

Four participants (*P1, P2, P3, P4*) described difficulties that limit the usefulness of dynamic analysis. These include certain anti-debugging or anti-tracing techniques, when portions of the program are missing, and when the binary's environment cannot be replicated (i.e., specialized or restricted hardware). In most cases, the reverse engineers are able to manipulate the execution environment by dynamically changing values or patching the binary to guide the program

down a specific path. As an example, malware frequently checks to see whether it is being run in a debugger, P2 simply patches the program to "automatically bypass that check." However, if the binary is incomplete or the system environment is not replicatable, then dynamic analysis rarely possible.

D. Documentation

The participating reverse engineers discussed three reasons for documenting results: personal note taking, internal team reporting, and external stakeholder collaboration. During the exploration of the binary executables, the participants document just enough information to be able to resume a task and rarely document the paths that were explored without success. The engineers described the following phases of documentation and current limitations in their process:

- **Copying-and-pasting**: To keep a record of their findings, reverse engineers copy and paste images (e.g., graphs) or textual representations of the binary executable into Microsoft PowerPoint, OneNote, or other electronic notebook software. These actions are purely *overhead*, disrupting the reverse engineer's workflow while manually duplicating information that the computer should be able to track automatically.
- **Writing notes**: Most early documentation is recorded using plain text files, creating small diagrams, and note saving. Notes provide some context for what the researcher is thinking at the moment, but they are not linked to the binary executable code to which they refer. For example, *P1* stated, *"I may record that at this [address] I saw something interesting, and may want to follow up later."* Microsoft Word or lightweight text editors such as notepad++, gedit, or vi were commonly used.
- **Organizing notes**: The problem with maintaining one central notes file is that it can become hard to search through. Conversely, the problem of maintaining many specialized files describing a system is that the collection may be difficult to sort through when compiling results. Like copying and pasting, the burden of designing a scheme to organize notes is purely overhead.

The participants expressed a general attitude that intermediate products such as hand-sketched diagrams and comments are "*ad hoc*," "*experimental*," or "*throw-away*." They noted that a significant amount of their work ultimately does not validate a useful hypothesis. One senior reverse engineer described, "*you go down a lot of dead ends, and you come up with a bunch of hypotheses. Eight out of 10 are dead ends" (P3)*. The same engineer also reported that he lacked a process to tell others, "*don't look here because I looked here and it's not useful. There are rarely remnants of dead ends*." Thus, the responses indicate that reverse engineers intentionally discard intermediate products when the end result does not seem insightful. However, recording these failed analysis paths may prove useful in communicating progress with team members, discussing encountered issues, or as a training tool to learn from past successes and failures.

After completing an analysis, the participants typically record their findings using Microsoft Word in a systematic manner, including a summary of findings, screenshots, diagrams, and recommendations. Translating the low-level findings into high-level results for consumption by non-technical stakeholders, including management, is a constant challenge. One engineer described this process: *"Typically, what I want to provide is evidence. So I take a lot of screenshots, [and] put them together with diagrams that document how this program works" (P5)*. Flow chart diagrams are most frequently developed in Microsoft PowerPoint or Visio and imported into the Word document. The length and quality of the reports vary based on the speed, goals, and complexity of the analysis:

> *One report we publish is called a Software Quick Look (SQL). It's a high-level description of the software that gives an understanding of what we think the software does, roughly how it works, and how it could be used on a particular weapon system. [The report] could be anywhere from 10 pages to 40 or 50 pages (P1).*

E. Collaboration

The participants reported meeting regularly with other colleagues to discuss long-term projects and immediate next steps. When working on large or complex projects, the participants reported an average team size of 2 - 5 people. During challenging tasks, sharing intermediate products with other engineers is considered a necessity. Indeed, all participants agreed that there are many times when it is necessary to ask for assistance or agreement on a finding. One engineer joked, *"You have a bunch of people that kind of know a little bit about reverse engineering scampering around semi-blindly trying to find cool things that are happening and then make sense of them" (P2)*. However, the participants suggest that reading each other's documentation alone is often not enough to understand the work that has been completed by somebody else: *"[I would] look at another engineer's notes, but probably only two or three things would make sense and then I'd still need to jump through [the code] to understand" (P5)*.

When the engineers are assigned to teams, they typically work in a co-located facility that allows for face-to-face communication among team members. The responses further revealed that the engineers do not typically apply specialized collaboration tools in their analysis, but rather rely on basic tools such as shared documents (hand-written or electronic), white boards, and, less frequently, a common knowledge repository. Hand-drawn diagrams were most often used to summarize program behavior or highlight specific insights

through flow graphs, sequence diagrams, and time-line graphs. *"There are huge needs for improved communication. It's really similar to team-source code development. You don't want to have multiple people working on the same part" (P4)*. The participants felt that the specialized collaboration tools were often too primitive to help them. These tools are designed only to provide a common space to collaboratively work together but do not effectively communicate hypotheses or insights during the sensemaking process.

V. Discussion & Implications

Based on an analysis of previous literature along with findings from our interviews, this section identifies opportunities for provenance support during reverse engineering analysis. In particular, we identify and describe opportunities where visual analytics support for analytic provenance will likely provide high value for practical needs. Provenance-based design aspects like provenance capture, provenance storage and retrieval, and provenance visualization needs to be designed with the identified tasks in mind. We believe that a strong understanding of the need for provenance support over particular tasks can guide design requirements to shape future research and development of provenance visualization and management tools. While these guidelines are drawn directly from our findings, further work is needed to validate their effectiveness.

Improved reverse engineering tools will likely always be needed to deal with the rapidly the changing software landscape. However, by analyzing the interview data we identified several potential tool concepts that could enhance reverse engineers' investigative and analytic capabilities. Provenance-based design decisions, such as provenance capture, provenance management, and provenance visualization need to be designed with the identified tasks in mind.

A. Provenance Capture

Analytic provenance captures the history and lineage of all of the actions performed by the user during the exploration process. Existing tools and prior research efforts in other heavily researched domains (e.g., intelligence analysis) have focused on supporting the capture of provenance data, but this not a solved problem. Heer and Agrawala [23] note that new visual analytic tools receive better user engagement and acceptance when they are integrated into the users' existing workflow tools. This restricts the selection of the domain-specific tools to ones with API's capable of capturing and storing fine-grained user activity.

Based on the interview data, the captured analytic provenance should be used to reproduce the binary manipulations that are performed by the engineer during analysis. Automated methods are imperfect, however, and there is a lack of clarity on the details of the provenance that needs to be captured. The reversing application and interfaces for future visual analytic systems need to be designed with provenance capture as a significant design focus. In a preliminary assessment of the reverse engineering tools described by the participants, Binary Ninja has the capability to automate the capture of user actions without significant external tooling, making it a potential leading candidate for provenance tool development.

Besides capturing user actions, capturing annotations while performing the analysis would provide important exploration process knowledge. Support for annotations in current reverse engineering tools usually consists of making comments assigned to particular instructions. These comments are usually limited to small hypotheses about the instruction or function under analysis. Some of the participants described the need for better methods for annotation management. Instead of the engineer referring to their handwritten notes, screenshots, or abbreviated comments in the reversing application, the provenance system can automatically record these data collections and the associated context in the program.

B. Provenance Management

When reverse engineers analyze compiled software, they must organize their findings, hypotheses, and evidence to *"tell a convincing story" (P4)*. In this context, the participants described challenges in keeping track of numerous code paths and data files, comparing the results of hypotheses with previously executed attempts, and remembering what they learned from past successes and failures. These problems persist because the current methods for managing and documenting findings in reverse engineering require too much user overhead and provide too little *context*. *Low overhead* means that the user can spend more of their time on applying program comprehension skills than on managing notes and findings. *High context* means that the user can directly correlate their past activities and results in the context of the reversing application.

During an analysis, the participants described the process of forming and testing hypotheses to make decisions based on the knowledge gained. This process is often iterative and can account for hours of exploration. Intermediate findings may lead to new strategies and decisions, resulting in a cyclic progression of knowledge building and understanding [5]. The reasoning steps are important to capture in order for the analysis to be explainable, reproducible, and trustworthy [24]. In particular, for analysis such as those for national security, these steps may be needed for criminal or intelligence investigations. Therefore, the accuracy and degree of detail of these reasoning steps are of critical importance.

While the participants report that not all hypotheses are proven useful, collecting and understanding failed hypotheses has proven to be of great value in other fields [25]. For instance, the provenance data can be used to better understand the strengths and weaknesses of the reverse engineer for both training and performance evaluations.

C. Provenance Visualization

The participants reported a pressing need for visualization tools to support their day-to-day workflow tasks, particularly during disassembly analysis. One engineer remarked on the cognitive challenges: *"All of the time in my head I'm trying to build up this graph of how this function calls this other function. We need better methods to keep track of the path activity" (P1)*. The other participants agreed, *"[We need] visualizations to help mentally layout and communicate the functionality of the binary" (P3)*. However, such tools must be flexible enough to support tasks with evolving objectives and needs.

As discussed in the findings, the participants' results are frequently documented by taking screenshots. However, static images cannot convey information about the exploration process. New provenance tools should not just communicate results, but also describe how these results were derived. The lack of an existing back-link from the results to the exploration stage and the underlying data makes it difficult (1) to reproduce and verify the findings explained in a report and (2) to extend the exploration to make new discoveries.

Reducing the time necessary to perform routine analysis activities was described by all of the participants. Early exploration tasks and the accompanying analysis generally require less than 24 hours for engineers to complete. The time available for a complete analysis depends on several factors, including workforce allocation and customer prioritization. Certain projects require a deep understanding of the software and are thoroughly analyzed with little time pressure. In such cases, *"analysis could take anywhere from six months to years for complex samples" (P1)*. In contrast, in situations where a quick response is necessary, the engineers will abandon a systematic approach for the sake of time. Without a systematic approach to provenance collection, however, reverse engineers risk missing important details and critical steps in their analysis.

Moreover, the participants described exploring software binaries as a time-consuming process. Indeed, a single analysis session can consist of hundreds of individual steps. After identifying interesting patterns, analysts may need to explore the relationships between them, generate potential hypotheses explaining those relationships, and find ways to verify the hypotheses. However, people have a limited working memory capacity and cannot hold all of these artifacts simultaneously. As such, they may forget previous findings and relationships, or remember but fail to retrieve the information needed. They may also forget how those findings were derived, making them more difficult to explain. Particularly for long and interrupted analysis sessions, the participants reported often becoming lost in the problem space: they are unable to examine their progress, unable to synthesize their discoveries, and unable to decide the next step effectively.

Visualizing the provenance history of user actions shows potential for addressing the challenges. Provenance data is often captured and visualized as state diagrams where the state of the data changes after each interaction. Frequently, the analytic provenance is visualized as graphs using nodes and branches [9], [26]. Branches represent pivot points where the analyst tried different hypotheses. When the volume of data captured is moderate, the data is intuitive and useful. However, supporting large volumes of data can become a challenge. Scalability should be considered for the design of provenance visualization interfaces.

VI. Limitations

A limiting factor was based on the nature of the participants' work. Due to confidentiality concerns, the researchers were not able to freely interview the participants about all aspects of their work or observe them in their daily work environment. This limitation was mitigated by focusing on their work needs and processes, not on specific protected tools, data, or sources. Related concerns also restricted the researchers' ability to use recording devices for three of the interviews. However, the risk of not adequately capturing all answers was mitigated by verifying the results with the participants.

External validity The biggest threat to external validity is that the captured responses may be atypical from other reverse engineering groups. The elicited requirements may be unique due to security limitations, mission goals, or resources of the participants, and therefore may not be representative of the larger population of reverse engineers. This threat is mitigated by adhering to a scripted protocol focusing on their workflow instead of specific techniques unique to their environment.

Internal validity The generality of the interview questions may be affected by researcher bias stemming from personal assumptions. This threat is mitigated by the pilot interview, the open-ended questions, and the follow-up discussions. The pilot interview helped the researchers refine the interview questions and ensure their relevance. Open-ended questions enabled participants to add or elaborate on their own opinions. Furthermore, the findings were confirmed in follow-up communications.

VII. Conclusions

This study explored the processes, tools, challenges, and visualization needs of software reverse engineers in order to inform the potential development of analytic provenance tools. Five workflow processes were identified: metadata analysis, disassembly analysis, dynamic analysis, documentation, and collaboration. Supporting the study's initial hypothesis, the participants in this study described a lack of adequate visualization and of a workflow supportive tool that contributes to an increase in analysis complexity, which

may also be prevalent in other reverse engineering settings. Analytic provenance tools may address this challenge by enabling the user to revisit the visualization states during the exploration process, validate hypotheses, organize findings, and present their process and results to others.

Many issues are currently known by the research community and supported by tools and techniques, but significant challenges remain for many tasks. This study reveals opportunities for improvement across critical reverse engineering tasks with limited current research and tool support. The results of this study suggest that capturing, managing, and presenting the exploration process are essential improvements for meeting the workflow needs of reverse engineers. Analyzing provenance histories may also contribute to the understanding of common analysis patterns and suggest enhanced interface designs.

Based on this study, the researchers are in development of the first analytic provenance system supporting reverse engineers' cognition. The prototype is currently being iteratively developed with the assistance of HCI and software reversing experts. A user-centered evaluation of the prototype will determine if the tool is effective in meeting the cognitive challenges identified in this preliminary work.

DISCLAIMER

The views expressed in this document are those of the authors and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government.

ACKNOWLEDGEMENT

The authors would like to thank the Air Force Research Laboratory's Information Directorate for their financial support. The authors would like to thank Vector 35 and developers of Visual Storytelling for their support.

REFERENCES

[1] P. D. O'Reilly, K. G. Rigopoulos, G. A. Witte, and L. Feldman, "2016 NIST Cybersecurity Program: Annual Report," National Institute of Standards and Technology, Tech. Rep. (NIST)-800-195, 2017.

[2] C. Treude, F. Figueira Filho, M.-A. Storey, and M. Salois, "An exploratory study of software reverse engineering in a security context," in *Proc. WCRE*, 2011, pp. 184–188.

[3] R. R. Klösch, "Reverse engineering: Why and how to reverse engineer software," in *Proc. CSS*, 1996, pp. 92–99.

[4] H. A. Müller, J. H. Jahnke, D. B. Smith, M.-A. Storey, S. R. Tilley, and K. Wong, "Reverse engineering: A roadmap," in *Proc. Future of Software Engineering*, 2000, pp. 47–60.

[5] A. R. Bryant, "Understanding how reverse engineers make sense of programs from assembly language representations," PhD Dissertation, Air Force Institute of Technology, 2012.

[6] R. Koschke, "Software visualization in software maintenance, reverse engineering, and re-engineering: A research survey," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 15, no. 2, pp. 87–109, 2003.

[7] J. Baldwin, P. Sinha, M. Salois, and Y. Coady, "Progressive user interfaces for regressive analysis: Making tracks with large, low-level systems," in *Proc. AUIC*. Australian Computer Society, Inc., 2011, pp. 47–56.

[8] H. M. Kienle and H. A. Müller, "The tools perspective on software reverse engineering: requirements, construction, and evaluation," in *Advances in Computers*. Elsevier, 2010, vol. 79, pp. 189–290.

[9] H. Stitz, S. Luger, M. Streit, and N. Gehlenborg, "AVOCADO: Visualization of workflow-derived data provenance for reproducible biomedical research," in *Computer Graphics Forum*. Wiley Online Library, 2016, pp. 481–490.

[10] C. Hargreaves and J. Patterson, "An automated timeline reconstruction approach for digital forensic investigations," *Digital Investigation*, vol. 9, pp. S69–S79, 2012.

[11] C. North, R. Chang, A. Endert, W. Dou, R. May, B. Pike, and G. Fink, "Analytic provenance: process+ interaction+ insight," in *Extended Abstracts on Human Factors in Computing Systems*. ACM, 2011, pp. 33–36.

[12] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen, "Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 31–40, 2016.

[13] J. Baldwin, A. Teh, E. Baniassad, D. Van Rooy, and Y. Coady, "Requirements for tools for comprehending highly specialized assembly language code and how to elicit these requirements," *Requirements Engineering*, vol. 21, no. 1, pp. 131–159, Mar. 2016.

[14] H. M. Kienle and H. A. Muller, "Requirements of software visualization tools: A literature survey," in *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*. IEEE, 2007, pp. 2–9.

[15] M. Ceccato, P. Tonella, C. Basile, B. Coppens, B. De Sutter, P. Falcarin, and M. Torchiano, "How professional hackers understand protected code while performing attack tasks," in *Proc. ICPC*, 2017, pp. 154–164.

[16] T. C. Summers, "How hackers think: A mixed method study of mental models and cognitive patterns of high-tech wizards," PhD Dissertation, Case Western Reserve University, 2015.

[17] P. Pirolli and S. Card, "The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis," in *Proceedings of International Conference on Intelligence Analysis*, vol. 5. McLean, VA, USA, 2005, pp. 2–4.

[18] G. Chin Jr, O. A. Kuchar, and K. E. Wolf, "Exploring the analytical processes of intelligence analysts," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 11–20.

[19] J. W. Creswell and V. L. P. Clark, *Designing and conducting mixed methods research*. Thousand Oaks, CA: Sage Publications, 2017.

[20] L. E. Wood, "Semi-structured interviewing for user-centered design," *ACM Interactions*, vol. 4, no. 2, pp. 48–61, Mar. 1997.

[21] M.-A. Storey, C. Best, and J. Michand, "SHriMP Views: An Interactive Environment for Exploring Java Programs," in *Proc. ICPC*. IEEE Computer Society, 2001, pp. 111–112.

[22] MaxQDA, "MaxQDA: Qualitative Data Analysis Software," https://www.maxqda.com/, 2019.

[23] J. Heer and M. Agrawala, "Design considerations for collaborative visual analytics," *Information visualization*, vol. 7, no. 1, pp. 49–62, 2008.

[24] N. Chinchor and W. A. Pike, "The science of analytic reporting," *Information Visualization*, vol. 8, no. 4, pp. 286–293, 2009.

[25] R. J. Heuer, *Psychology of Intelligence Analysis*. Lulu.com, 1999.

[26] Y. B. Shrinivasan and J. J. van Wijk, "Supporting the analytical reasoning process in information visualization," in *Proc. CHI*. New York: ACM, 2008, pp. 1237–1246.