An Investigation of Interaction and Information Needs for Protocol Reverse Engineering Automation

Samantha Katcher[*]
Department of Computer Science
Tufts University
Medford, Massachusetts, USA
MITRE Corporation
Bedford, Massachusetts, USA
samantha.katcher@tufts.edu

James Mattei[*]
Department of Computer Science
Tufts University
Medford, Massachusetts, USA
james.mattei@tufts.edu

Jared Chandler
Department of Computer Science
Dartmouth College
Hanover, New Hampshire, USA
Jared.D.Chandler@dartmouth.edu

Daniel Votipka
Department of Computer Science
Tufts University
Medford, Massachusetts, USA
daniel.votipka@tufts.edu

Abstract

Protocol reverse engineering (ProtocolREing) consists of taking streams of network data and inferring the communication protocol. ProtocolREing is critical task in malware and system security analysis. Several ProtocolREing automation tools have been developed, however, in practice, they are not used because they offer limited interaction. Instead, reverse engineers (ProtocolREs) perform this task manually or use less complex visualization tools. To give ProtocolREs the power of more complex automation, we must first understand ProtocolREs processes and information and interaction needs to design better interfaces.

We interviewed 16 ProtocolREs, presenting a paper prototype ProtocolREing automation interface, and ask them to discuss their approach to ProtocolREing while using the tool and suggest missing information and interactions. We designed our prototype based on existing ProtocolREing tool features and prior reverse engineering research's usability guidelines. We found ProtocolREs follow a flexible, hypothesis-driven process and identified multiple information and interaction needs when validating the automation's inferences. We provide suggestions for future interaction design.

CCS Concepts

• **Security and privacy** → **Software reverse engineering**; • **Human-centered computing** → **Interaction design**; **Human computer interaction (HCI)**.

ACM Reference Format:
Samantha Katcher, James Mattei, Jared Chandler, and Daniel Votipka. 2025. An Investigation of Interaction and Information Needs for Protocol Reverse Engineering Automation. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26–May 01, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3706598.3713630

1 Introduction

There has been a long history of developing automation for reverse engineering, with significant research and industry investments made [3, 5, 19, 21, 32, 50, 63, 70, 74, 78]. While these efforts have made significant strides, human intelligence is required to supplement automation and will likely remain necessary for the foreseeable future [11, 57, 77, 79]. Therefore, recent research has focused on understanding and supporting reverse engineers' processes [11, 46, 75, 77]. However, there is limited human-automation interaction research for reverse engineering, with existing research focused on existing automation's usability flaws [47], challenges in automation setup [59], or developing novel AI/ML techniques without primarily considering usability [2, 4, 7, 35, 43].

In this paper, we take a first step toward closing this gap. Because reverse engineering is a complex domain, we expect variation in information and interaction needs for specific tasks. Therefore, we first focus on a specific reverse engineering sub-field, *protocol* reverse engineering (ProtocolREing). To our knowledge, prior work investigating reverse engineering's human factors has almost exclusively considered *software* reverse engineering (SoftwareREing), with none studying ProtocolREing. ProtocolREing is the task of inferring a protocol, i.e., structured message sequence where each message segment serves a specific purpose, from a stream of unstructured data [39]. This task is most commonly performed for malware analysis, where an analyst reviews data transmitted between a bot and bot controller to determine what is transferred to write signatures for malware identification and potentially manipulating these communications. ProtocolREing is also used when investigating the software is challenging. For example, embedded systems often have unique architectures that are not supported by existing disassembly and emulation tools. ProtocolREing can be thought of as attempting to identify patterns in unstructured data, where protocol reverse engineers (ProtocolREs) must consider many possible protocols and determine which fits the data. This differs from SoftwareREing, where software reverse engineers' (SoftwareREs)

[*] Both authors contributed equally to this research

ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3713630

task is to interpret a sequence of program instructions with known meaning to determine their function. It is worth noting that ProtocolREing is closer to problems in data science where an analyst investigates data records to infer some relationship. However, in ProtocolREing, there are clear logical and structural relationships between fields in the underlying protocol, restricting the search space of possible protocols. For example, messages are typically divided into fields, multiples of a byte in size.

ProtocolREing's ambiguity introduces a unique challenge and an opportunity for automation to help ProtocolREs navigate this data. Several tools have been developed to automate ProtocolREing [13, 36]. However, these tools offer limited interactivity only through configuration file changes or modifications to the automation's codebase. This requires users to be familiar with the automation's design, creating a high barrier for interaction. Conversely, there are full-featured user interfaces for manual ProtocolREing, most notably Wireshark [25]. However, these tools only support limited automation for matching data to known protocols (e.g., TLS, HTTP). Because of the unique task characteristics and dearth of existing tools, we sought to understand the interaction needs of ProtocolREs to guide the development of the next-generation automation.

Specifically, we seek to answer the following research questions:

- RQ1: What process do ProtocolREs follow?
- RQ2: What information do ProtocolREs need from ProtocolREing automation?
- RQ3: What interactions should ProtocolREing automation support?

Because there is no existing literature on ProtocolREs, we sought to understand ProtocolREs' processes, e.g., what types of questions they ask about the data, what hypotheses they generate, the specific steps taken to learn more about the data, and what information informs their decisions. We also wanted to understand their unique information and interaction needs throughout.

To answer these questions, we drew inspiration from Votipka et al.'s retrospective observational interviews with expert SoftwareREs to understand their process when investigating unfamiliar programs [76]. As there is no theoretical basis on which to build quantitative assessments, we pursued an exploratory qualitative approach drawing from prior work in expert decision-making [12, 40].

We conducted 16 interviews with ProtocolREing professionals, asking them to interact with a paper prototype interface with automated features. We asked participants to consider sample tool output from the automation and provide feedback on remaining information needs, missing interactions, and thoughts on how automation can support or hinder their process.

We found that generally, the ProtocolREing process model centers around determining potential field boundaries and datatypes. ProtocolREs cyclically update their hypotheses about field boundaries and datatypes, which are tightly coupled. ProtocolREs incorporate information from various sources, including context research about the application (if available) and any tooling support. ProtocolREs typically operate within field boundaries, occasionally comparing a given field against its neighboring fields to validate or challenge their current working hypothesis. Above all, ProtocolREs need the ability to modify field boundaries, datatypes, and other configuration options quickly and easily as they cycle quickly between hypotheses, testing whether protocols that fit one part continue to fit when looking at other message segments. Any tooling or automation inhibiting this quick, cyclic process is highly disruptive to ProtocolREs. This differs from SoftwareREing, which has been shown to have a natural break between identifying a hypothesis and testing that hypothesis, as the analyst shifts from static to dynamic analysis.

Due to the application-specific nature of ProtocolREing and the inherent ambiguity of the task, automated results should be conservative and easy to reverse, acting as a guide to an iterative search. This follows from the observation that the process was highly iterative as ProtocolREs try multiple possible protocol options before determining the correct ones. This could be seen in our participants' nearly universal belief that automation could not correctly predict an entire protocol, but was instead useful for providing suggestions for parts of the protocol. Instead of taking the suggestion as an end product, they wanted interactions that allowed adjusting the automation's assumptions to refine the analysis and quickly validate any suggestions iteratively. This suggests future work in human-automation interaction is particularly important for ProtocolREing as human review is necessary. Without good interaction design, the automation is unlikely to be useful.

Finally, because the ProtocolREing process is mostly focused on iterating possible datatypes and field boundaries, we observed participants' requests for additional information and interaction were shaped by these two protocol properties. This included providing datatype definitions as a way of restricting automated suggestions or navigating the search space by investigating neighbor field boundaries.

From these results, we offer suggestions for ProtocolREing tool developers and future research.

2 Background and Related Work

To our knowledge, no work has focused on understanding the process and automation interaction needs of ProtocolREs. Therefore, we begin by describing ProtocolREing and existing work on automated techniques for ProtocolREing. Then, we describe the literature on human factors of reverse engineering more generally.

2.1 Defining ProtocolREing

ProtocolREing is the task of recovering an abstract description of the type, function, and sequence of data, i.e., a protocol, which provides meaning to a sequence of bits. This task often takes the form of analyzing the messages transmitted between systems to understand what is being communicated. While ProtocolREing also can be performed to infer a file format, we will discuss it in the context of network communications for simplicity.

We now walk through the example used in our study to provide a concrete picture of ProtocolREing. Imagine an analyst is given the three hex messages shown at the top of Figure 1 in part A. These messages represent network packets transmitted from an air traffic control broadcasting station relaying information about in-progress flights. Knowing the broadcast protocol could be useful for security analysis, e.g., to support structured fuzzing-based analysis to identify vulnerabilities, or other data processing to track the information transmitted. It is possible to capture this information by inspecting the software's instructions, i.e., SoftwareREing; however, this is often challenging in legacy or embedded systems (such as an air traffic control system, medical devices, or automotive equipment) that use unique or no longer common architectures. Popular disassemblers or emulators do not support many of these architectures, making analyzing or running the code challenging. However, capturing network traffic is simpler and more generally applicable. ProtocolREing is also common in malware analysis, where code obfuscation and anti-debugging techniques make SoftwareREing difficult or impossible.

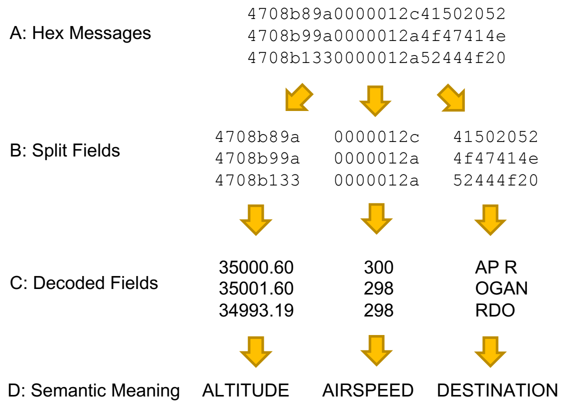


Figure 1: Example hex messages (A) are examined by an analyst. The analyst splits the messages into three fields (B) based on similarities and differences in the data. Each field is then decoded according to an inferred type (C) with the resulting semantic meaning shown as (D).

No matter the scenario, ProtocolREs seek to reconstruct the message format and uncover each field's semantic meaning. They must determine the meaning of these messages by identifying which portions of each message belong to discrete fields (e.g., byte boundaries), the fields' datatypes (e.g., float, integer, ASCII), and ultimately, each field's semantic meaning in the application's context.

Returning to our air traffic control example, Figure 1.B shows the ProtocolRE should divide each message into three fields, each of 4 bytes. Then, in Figure 1.C, these fields need to be converted into their appropriate data types: a 32-bit IEEE Float, a 32-bit integer, and a four-character ASCII field, respectively. They arrive at these types by observing the range of values for each field and identifying patterns associated with different datatypes. For example, the ASCII field contains only values within the standard human-readable ASCII range, while the Float field contains values mostly outside that range. Similarly, the leading null bytes in the second field indicate an integer where the most significant bits seldom change.

Finally, the ProtocolRE must infer what these fields represent. They would examine the floating point, integer, and ASCII values and infer they are related semantically to aircraft. The ProtocolRE might note the ASCII values are consistent with airport destinations, the floating point values fit aircraft cruising altitudes, and the integer values match aircraft speeds. These semantic meanings are shown in Figure 1.D. The information describing field boundaries, data types, and semantic meanings comprise the protocol definition, which could be used to interpret future messages consistent with this protocol.

Our air traffic control scenario illustrates the challenges associated with ProtocolREing due to the various datatypes used in a single message and the lack of source code to validate findings. For this scenario, a ProtocolRE must work exclusively with the observed messages and could not rely on access to the aircraft, the base station, or the software running on either.

ProtocolREing compared to SoftwareREing. Because significant prior work has investigated SoftwareREing, a natural next question is whether ProtocolREing is sufficiently different in how people perform the task, as both seek to infer function from program data. The difference is in the data's ambiguity. In SoftwareREing, program instructions with a specific function are investigated. It is the SoftwareRE's task to interpret a sequence of instructions according to their specification to determine their function when performed in the given sequence. In contrast, ProtocolREs attempt to determine meaning from unstructured data. In our air traffic example, the ProtocolRE might try splitting the first field into two 2-byte fields instead of one 4-byte field. This changes possible data types that could be used to decode the data and would impact the perceived semantic meaning. Because there is no ground truth, the ProtocolRE makes decisions about the data based on their *most likely* interpretation and may adjust those opinions later. Therefore, ProtocolREing is performed over many messages to identify trends that reveal underlying structures.

2.2 Existing tools to support ProtocolREing

Manual ProtocolREing is challenging and time-consuming, especially as ProtocolREs review larger data volumes. Several tools exist to help ProtocolREs visualize message data and perform common ProtocolREing tasks.

One of the most well-known tools for examining network data is Wireshark [25]. Wireshark is used to analyze packets' sequences and examine their values. Wireshark allows a ProtocolREs to filter packets by format and packet field values for known protocols (e.g., TCP, HTTP). A ProtocolRE can then view both the raw binary packet representation as hexadecimal data, the packet decoded as text, and the individual packet fields interpreted when the packet is known to belong to a specific protocol. However, Wireshark does not provide interactive support for users attempting to determine the specification of an unknown protocol. Instead, users must determine the protocol separately and write a parser to be applied by Wireshark. Additionally, Wireshark is generally limited to viewing only one packet at a time, making between-packet comparisons difficult.

Another popular online tool supporting ProtocolREing is CyberChef [26]. CyberChef is a web application with several ProtocolREing-relevant features, such as encoding or decoding Base64, Hex conversions, hash and checksum calculations, and IPv6 header parsing. While CyberChef allows users to manipulate data without managing any custom algorithms, it offers limited automation or guidance as to possibly appropriate data transformations.

There exists significant academic work developing novel ProtocolREing interfaces. Conti et al. presented a broad survey using visualizations to support security goals, including ProtocolREing [15, 17]. However, these visualizations show data distributions instead of directly presenting messages. Other examples of these visualizations incorporate temporal elements and multiple file streams [1, 16].

An alternative approach comes from methods for investigating binary files. These methods focus only on visualizing a single file of unknown format. Conti et al.'s BytePlot interface is one example [17]. BytePlot represents the same data with different visualizations, each tuned to various aspects and reverse engineering process functions. For instance, one visualization provides an overview of all file binary data, while another visually summarizes individual regions' byte-value distribution.

We leverage the features and designs of these tools as inspiration for the prototype we used in our interviews as described in Section 3.1.

Automation for ProtocolREing. A growing body of literature and development effort is directed at automation support for ProtocolREing. There are various available tools with autonomous features [22, 39, 54]. Broadly, ProtocolREing automation is classified into two categories: tools that analyze captured network messages exchanged between two components or tools that leverage source code or binary files from the application to uncover the protocol structure. For simplicity, we focused on the former for the paper prototype.

Automation in this context attempts to identify three things: message formats, field semantics, and rules governing the sequence of exchanged messages. Several recent advancements in automated field boundary detection leverage intrinsic repeated structures or distributions of values throughout messages. Kleber et al. utilize Sokal and Michener's bitwise similarity metric to track the change in congruence between bit values [38, 67]. This metric allows their tool NEMESYS to select candidate field boundaries by identifying inflection points at the rising edges of this congruence metric. Luo et al. proposed using Latent Dirichlet Allocation models to characterize message types and n-grams [8, 45]. This information would be used for clustering algorithms to group similar fields from messages that likely shared a type. Similarly, Ye et al. utilized multiple-sequence alignment to justify all messages before using probabilistic inference to cluster messages by their most likely candidate type [80]. Chandler et al.'s BinaryInferno leverages an ensemble of specialized detectors to identify common data types and serialization patterns [13]. BinaryInferno identifies field boundaries by looking for all semantic types consistent with the message bytes and then finding the overall description with the highest explanatory power.

Although much prior work has developed novel automation techniques for ProtocolREing, these tools do not support user interaction beyond complex modification to configuration files or editing the automation's code. Therefore, it is unclear how these tools can best provide inferences to ProtocolREs to help them analyze large volumes of messages and support task efficiency. Our study investigates how to close this gap between functional interfaces and automation, empowering the next generation of automation for ProtocolREing.

2.3 User Studies with Reverse Engineers

To our knowledge, no prior studies have investigated the human factors of ProtocolREing; a growing body of research has studied the related tasks of SoftwareRE and hardware reverse engineering. Specifically, this work has studied reverse engineers' processes and the usability of available tools in these contexts.

Investigating SoftwareREs' processes. SoftwareREs were perhaps first studied by Bryant, who conducted interviews with four professional SoftwareREs and developed a sense-making model of the SoftwareREing process [9]. Votipka et al. built on this work, conducting 16 retrospective observational interviews with expert SoftwareREs investigating an unfamiliar program [75]. Votipka et al. refined the SoftwareREing process model into a three-phase iterative model. They observed SoftwareREs typically began with an overview phase, where they sought to identify code segments of interest for further review. During this phase, SoftwareREs run the program and observe its behavior, identify unique strings or APIs used, and specific functions to investigate further. Next, SoftwareREs consider the segments identified in the subcomponent scanning phase, reviewing them to generate hypotheses and specific questions. SoftwareREs scan for specific beacons and unique data-flow or control-flow paths. Finally, they test their hypotheses in the focused experimentation phase. In this phase, they execute the program or mentally simulate it line-by-line to produce specific answers for a certain input. Note, each of these phases is treated as a distinct step, using different methods for each and dividing hypothesis generation (overview and subcomponent scanning phases) from testing (focused experimentation phase). Additionally, SoftwareREs often switch types of analysis between the subcomponent scanning and focused experimentation phases, i.e., static and dynamic analysis, respectively. SoftwareREs typically iterate between the subcomponent and experimentation phases, continually learning more about the program until they can answer their target questions, such as identifying a vulnerability or understanding a malware infection. This model has subsequently been supported in large-scale quantitative measures showing how SoftwareREing experience influences analyst actions and strategies [11, 24, 46]. Additionally, Becker et al. investigated another reverse engineering field, *hardware* reverse engineering (HREing), looking at HRE's technical and cognitive processes. They observed a similar three-phase model for this sub-discipline of reverse engineering [6]. Although the specific context differed from ours, we expected ProtocolREs might have similar processes. Therefore, we utilized the process model and design guidelines proposed from this prior body of research for our paper prototype design (see Section 3.1) to ensure the presented interfaces fit the users' expected workflow.

Studies of SoftwareRE tool usability. Most related to our work, recent studies have assessed the usability of specific tools and techniques for SoftwareREing. Using the guidelines proposed by Votipka et al., Mattei et al. assessed the expected usability of 288 SoftwareREing tools [47]. Ploger et al. studied the usability challenges for beginners when using a fuzzer and static analyzer, showing that setting up these tools was a significant barrier to their use [59]. Shoshitaishvili et al. developed the HaCRS system, which created an interface allowing users to provide input seeds to mutational fuzzers without needing expertise with fuzzers [63]. They demonstrated that non-experts' input improved tool outcomes but did not assess the usability of this interaction. Finally, Yakdan et al. developed DREAM++, a usability-optimized decompiler leveraging heuristic-based transformations to improve decompiled code readability [78]. DREAM++ achieved concrete performance gains by improving decompiled code readability in a user-informed manner. These studies solely focus on designing and implementing visualizations of single binary files. In contrast, we focus on understanding ProtocolREs' processes when investigating a collection of multiple network packets. We draw inspiration from these studies to design our prototype and assess what information and interactions ProtocolREs need.

3 Methods

We conducted 16 semi-structured interviews with ProtocolREs with significant ProtocolREing experience. We asked them to interact with a paper prototype of an automated ProtocolREing tool to see how they approach ProtocolREing tasks and how they take information from and interact with automation. Interviews were conducted between February and June 2023. We describe the development of our prototype (referred to as the Prototype Interface), mock use case, and interview protocol, as well as our study recruitment, analysis approach, ethical considerations, and our study's limitations.

3.1 Prototype Interface Design

To shape the interview discussion and provide a reference point, we presented participants with a paper prototype to understand how ProtocolREs interact with automation. A paper prototype is beneficial as it minimizes distractions caused by interface aesthetics and technical issues, ensuring the participants' focus was on core information needs and user interactions [64].

To produce a realistic interface, we first examined existing ProtocolREing tools and ProtocolREing process literature to identify common features. Then, we defined design guidelines based on prior HCI and SoftwareREing literature. This section describes our feature and guideline development and how we produced the final components of our prototype, which we refer to as prototype views.

3.1.1 Tool review. As a preliminary step, we determined the features commonly provided by existing tools to ensure a realistic interface. We leveraged one author's ten-plus years of experience with ProtocolREing tool development to identify a preliminary list of ProtocolREing tools. Then, we broadly surveyed the web using an online keyword search for other available tools. We searched the terms "reverse engineering" and "Protocol reverse engineering". We combined them with "frameworks," "tools," and "networking." We searched every resulting link, webpage, or video for downloadable tools until we repeatedly encountered duplicate tools. For each search, we reviewed at least the first page of results.

From our search process, we found a variety of tools broadly categorized as Hex Editors (ImHex, Hexinator, ReHex, Synalysis, and 010 Editor), Interactive analysis tools (Cyberchef), Network analysis tools (BinaryInferno, Nemesys, Netplier, Netzob, and Wireshark), and data description tools (Kaitai). These tools are listed in Table 1. Due to our extensive search process, we expect our resulting tools to represent what analysts would be familiar with through their day-to-day functions to assist with ProtocolREing. To confirm this, we asked participants in our interview to state tools they previously used for ProtocolREing. Almost all the tools mentioned had overlapping features with those identified by our search. The exception was tools for packet generation (i.e., Scappy, Fiddler, and FakeNetNG), which went beyond direct message analysis and were used to produce additional message data for testing. We discuss these in detail when describing participants' ProtocolREing processes (see Section 5.2).

3.1.2 Feature review. We reviewed each tool for the features they provide for ProtocolREing. This review identified the following features observed in at least one tool. Table 1 indicates the features offered by each tool.

F1 **Hex** - All our surveyed tools present raw data first in hex. This offers a view into the raw data in a compressed format, i.e., instead of showing each bit.
F2 **Decode Type** - Many tools allowed users to convert the hex data to various other datatypes, either one datatype or multiple simultaneously.
F3 **Set Type** - Some tooling allowed users to specify and restrict fields to a certain type. This was a common feature of several tools.
F4 **Suggest Type** - When working on ProtocolREing, there is missing information about the structure and type of the messages. The tool can offer users suggested types based on the raw hex values to guide their process.
F5 **Suggest Field** - Some tools try to infer potential field boundaries from the data and present these candidate fields to the user.
F6 **Data Distribution (Stats)** - Another method for ProtocolREs to glean insights about the raw data is through visualizations and summary statistics. Tools can present this information visually with graphs or via text.
F7 **Multiple Data comparisons (Compare Multiple)** - All tools can simultaneously operate on multiple pieces of data, packets, files, or streams, allowing for comprehensive comparisons and analyses. For example, several tools support the incremental development of data parsers to handle various input streams.
F8 **Filter/Sort** - Many tools offered filtering or sorting functionality across hex messages. ProtocolREs could use these to sort by an identified sequence number to preserve inter-message contexts.
F9 **User Scripting** - Most of the tools could support extended functionality from user-provided scripts.

In addition to the features identified in existing tools, we incorporated additional opportunities for automation interaction. While some tools offer automated support, none allow ProtocolREs to interact with the automation directly through the user interface. Instead, users could only impact the automation by changing a configuration or providing a new template. These are speculative

	Hex	Decode Type	Set Type	Suggest Type	Suggest Field	Stats	Compare Multiple	Filter /Sort	User Scripting	Use Case
BinaryInferno [13]	✓	-	-	✓	✓	-	✓	-	-	Network
Cyberchef [26]	✓	✓	✓	✓	-	✓	✓	✓	✓	Binary Data
ImHex [34]	✓	✓	✓	✓	-	✓	✓	✓	✓	Binary Data
Kaitai [36]	✓	✓	✓	✓	-	-	✓	✓	✓	Binary Data
Hexinator [33]	✓	✓	✓	✓	-	✓	✓	✓	✓	Binary Data
Nemesys [65]	✓	-	-	-	✓	-	✓	-	-	Network
Netplier [80]	✓	-	-	-	✓	-	✓	-	-	Network
Netzob [55]	✓	✓	✓	✓	✓	✓	✓	✓	✓	Network
ReHex [14]	✓	✓	✓	✓	-	-	✓	✓	✓	Binary Data
Synalysis [71]	✓	✓	✓	✓	-	✓	✓	✓	✓	Binary Data
Wireshark [25]	✓	✓	-	✓	-	✓	✓	✓	✓*	Network
010 Editor [66]	✓	✓	✓	✓	-	✓	✓	✓	✓	Binary Data

Table 1: Table of features and capabilities of various ProtocolREing tools. The * notation indicates a tool supports the feature via plugins.

features not provided by prior tools but essential to investigating issues of human-automation interaction.

F10 **Split-merge fields** - We included the ability to split and merge fields at every byte boundary and automated suggestions for additional field boundary splits beyond the preliminary fields. This additional flexibility allows users to explore the data more meticulously and more easily change their field boundaries as they uncover more information about the data.

F11 **Alternative data type and field boundary suggestions** - We provide additional possible suggestions besides the preliminary suggested type and field boundaries. This emulates automation, providing the top N options instead of just the option with the most confidence. This allows users to consider several suggestions in parallel and may prompt users to consider potentially overlooked options [83, 84].

3.1.3 Design guidelines. Next, we describe the design guidelines that directed the development of our Prototype Interface. We drew on two bodies of literature to determine these guidelines. First, we adopted recommendations from prior work in SoftwareREing tool usability [47, 76]. We also consider guidelines from exploratory visual analysis [61, 62] From these reviews, we established the following guidelines:

D1 **Filter, Zoom, and Details on Demand.** Prior literature has highlighted the importance of user-defined focus and scope. For our Prototype Interface, this was filtering the data or zooming and drilling down to get a more detailed view [62]. Tools should allow users to get a richer, more detailed view of their data as they need it. These core functionalities enable users to refine their focus and extract only the relevant details to continue investigating their hypotheses.

D2 **Input and Output presented in line** - To integrate into users' processes more seamlessly, tools should accept inputs and present outputs in the same window or medium with which the user is engaged. For our context, we ensured that users had potential input and output options for each interface view, which were presented in line with the data.

D3 **Analysis Tuning** - To better suit the needs of their users, tools should allow their functionality to be customized or fine-tuned. This additional flexibility enables users to impart contextual information and better interpret the results of any automated tooling.

D4 **Readability Improvements** - Wherever possible, tooling should support additional readability improvements to allow users to navigate their environment and data more efficiently. This can take many forms, including color coding, data distribution or entropy visualizations, and decoding data types.

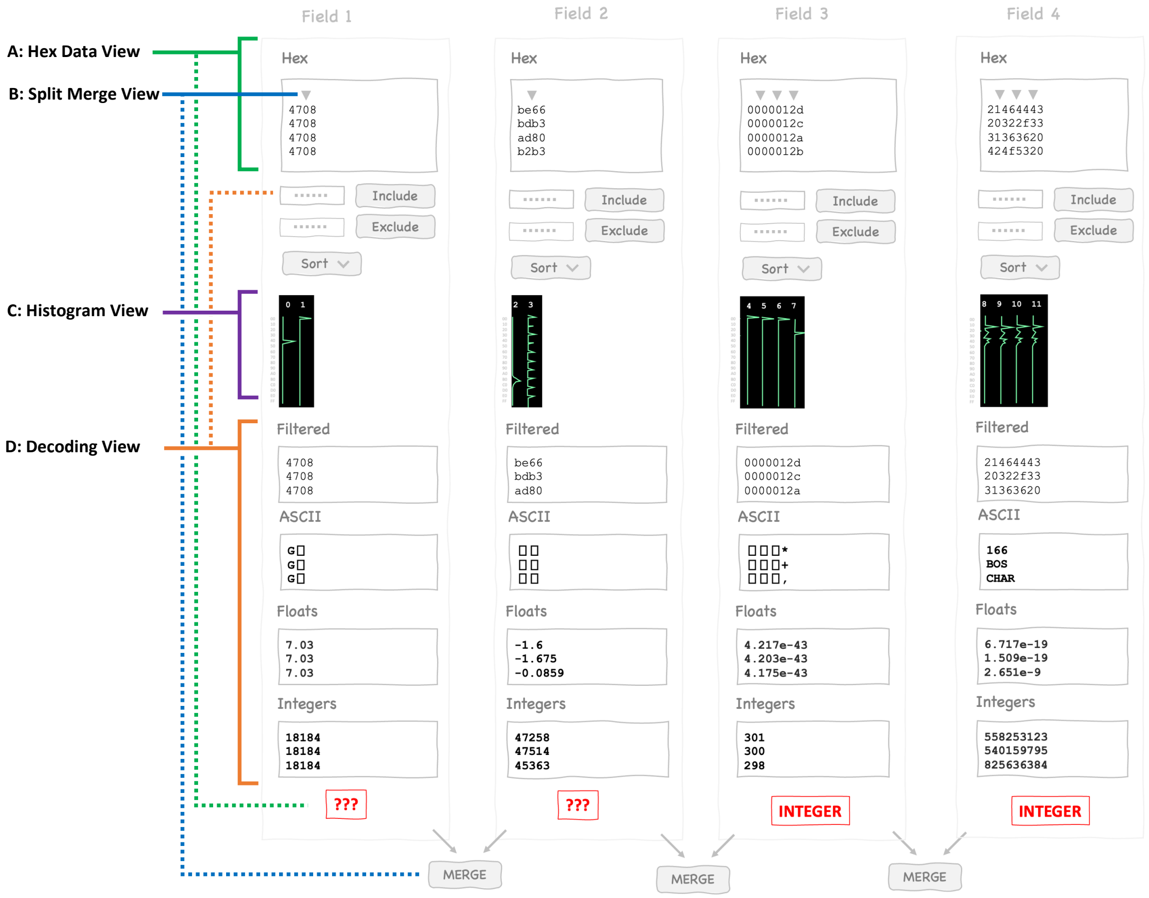
3.1.4 Prototype Views. We created five distinct views for our Prototype Interface to include many of the features identified in our tool review (Section 3.1.1. Each view was developed with our design guidelines in mind by a research team member with over ten years of professional graphic and web design experience. Here, we describe each in turn.

Hex View (F3, F4, F1, F7, F5, F11, D2, D3). The first view consists of showing multiple related messages simultaneously, as shown in Figure 2.A. Intuitively, each message is shown on a line, with the bytes aligned across messages using a mono-space font in Hex (F1, F7, D2). This format is consistent with ASCII text display, with data ordered left to right, top to bottom.

This interface automatically assigns field boundaries to the raw data and separates the data into candidate fields and suggested types, as shown in Figure 2.A (F4, F5). Users can select from alternative data type suggestions the automation has deemed potentially valid (F3, F11, D2, D3). This broad view allows users to get an overview of patterns across all the data or narrow their focus to a single field boundary, type, or message.

Split-Merge View (F3, F4, F1, F7, F5, F10, F11, D2, D3). The second view is similar to the first in how the data is presented (F4, F1, F7, F5, D2), but it gives more control to the user. Users can modify the suggested initial field boundaries, combining fields or splitting to create new boundaries and change the suggested types (F3, F10, D3). Users can adjust field boundaries via a graphical interface presented directly in line with the hex data immediately (D2), as shown in Figure 2.B.

In addition to the initially suggested field boundaries, this interface allows the user to choose alternative field splits using a



	Field 1	Field 2	Field 3	Field 4
Hex	4708 4708 4708 4708	be66 bdb3 ad80 b2b3	0000012d 0000012c 0000012a 0000012b	21464443 20322f33 31363620 424f5320
Filtered	4708 4708 4708	be66 bdb3 ad80	0000012d 0000012c 0000012a	21464443 20322f33 31363620
ASCII	G□ G□ G□	□□ □□ □□	□□□* □□□+ □□□,	166 BOS CHAR
Floats	7.03 7.03 7.03	-1.6 -1.675 -0.0859	4.217e-43 4.203e-43 4.175e-43	6.717e-19 1.509e-19 2.651e-9
Integers	18184 18184 18184	47258 47514 45363	301 300 298	558253123 540159795 825636384
	???	???	INTEGER	INTEGER

Figure 2: Combined View interface shown to participants during the interview. The labeled elements correspond to the components shown for each specific view, where were individually introduced progressively, building the final Prototype Interface. That is, each specific view could be generated by only showing the part of the image marked with the associated bracket or dotted line. A - Hex View, B - Split-Merge View, C - Histogram View, D - Decoding View

wedge glyph. Similar to the additional suggested data types, this indicator can be a subtle nudge to ProtocolREs to explore other field boundary and data type possibilities when investigating the data (F11). One of the wedge glyphs was highlighted to show a candidate field boundary the automation thought was possible. We used this to ask participants how they preferred incorporating automated tooling.

Histogram View (F1, F6, F7, D1, D4). Another way to visualize the data is through a histogram, as shown in Figure 2.C. The visualization is organized by byte position, with each column representing the aligned bytes across the collection of Hex messages (F1, F7). This visualization shows the frequency of different values at a byte offset into a message as a line chart (F6, D1, D4). We order the chart with lower hex values at the top and higher hex values at the bottom. A legend of hex values is shown on the vertical axis. Visualizing the data in this manner allows the frequency of values to be compared within a byte offset or between fields. We removed the suggested types and split-merge capabilities from the prior interfaces to avoid overloading participants with too many new features simultaneously.

Decoding View (F1, F7, F2, F8, D1, D3, D4). Our penultimate view introduces several key features to manage the data: decoding as multiple data types (F2), filtering, and sorting (F8, D1, D3). First, this view showed the data decoded as a variety of standard datatypes, as shown in Figure 2.D. This interface allows the analyst to observe the decoded values and judge whether any hex values translate to valid values for multiple common data types (F1, F7, F2, D4). By providing multiple comparisons, this view facilitates exploration and rapid evaluation of hypotheses for data types where the interpreted values appear random or inconsistent with analyst knowledge.

This view allows analysts to filter or sort messages to refine their focus on specific data (F8, D1, D3), as shown in Figure 2.D. We did not restrict how filtering or sorting could be utilized but instead asked participants how they might want to perform these operations. For example, they could modify their focus based on criteria such as timestamps, source and destination addresses, or message content. Because we tried to cover several features but wanted to ensure we were respectful of our participants' time, we could not always cover all views when participants gave more detailed answers earlier in the interview. To avoid missing participants' responses to the combination of all views, we chose to skip the filtering and sorting, which are well-established in UI design practice [23, 51, 56, 61, 72] and already employed in several existing tools in Figure 1. The interviewer decided whether to skip filtering and sorting at the time of the interview. These features were skipped in six of the 16 interviews.

Combined View (F1-F8, D1-D4). The final view integrates elements of all prior views as shown in Figure 2. The Combined View allows candidate fields to be split and merged(F4, F5, D3, D4), interpreted as various data types (F3, F1, F7, F2, D1, D2, D3, D5), and displays summary visual information(F6, D1, D2, D5). The advantage of this interface is it allows simultaneous comparison across multiple dimensions and gives users multiple options to tweak the data and explore hypotheses. Participants were prompted to explain which interface elements are useful, overbearing, or distracting while performing their tasks.

Due to the nature of our Prototype Interface, we chose not to directly implement a scripting view across our interface. Instead, we prompted participants during the interview to describe how they would like to extend each view with scripting or how they would like to interact with the combined interface programmatically.

3.2 Interview Protocol

Interviews were conducted and recorded on Zoom and lasted 60 minutes on average. The same author conducted all interviews to ensure consistency. Participants were asked to review each view and provide feedback. Our study protocol is summarized in Figure 3, and the script and additional interview materials are provided in an OSF folder [37].

Participant background and experiences in ProtocolREing (Figure 3.A,B). The interviews began with an open-ended question asking participants to describe their ProtocolREing backgrounds and expertise. We concluded this section by asking participants to describe ProtocolREing tools they commonly use.

Interview scenario. Participants were told to determine the protocol of an example set of captured messages. We used the example given in Section 2.1 to populate our interface but did not provide context to participants. The interview started with a sample of the message data shown as hex, with one message per line. We instructed participants that each message was transmitted in the same direction (host A -> host B), so each line was a different instance of one-directional communication.

View interaction (Figure 3.C-G). Next, we showed participants the different views described in Section 3.1.4. Participants were progressively walked through the different views in the following order: Hex View, Split-Merge View, Histogram View, Decoding View, and finally, Combined View.

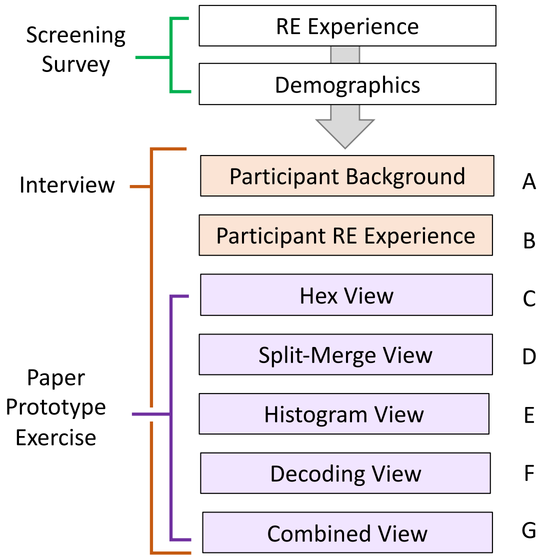


Figure 3: Study recruiting and interview structure.

We asked participants to walk through their approach to investigate this data using all of our prototype views. For each view, we asked participants to review the automation output and say if they had any hypotheses about the data, if they thought any output was incorrect, and what other features or information they would need to complete their process. We asked participants to "think aloud" to reveal their ProtocolREing approaches and decision-making criteria. We asked follow-up questions to investigate participants' thought processes, information and interaction needs, or any sources of confusion. To challenge participants' perspectives, we introduced an incorrect automated suggestion across all views and added a noisy errant message during the Split-Merge View.

The interview scenario was developed independently of the design of the Prototype Interface. While we consistently used the airline data scenario for testing, the interface was designed to be versatile and applicable to many other protocols beyond just this use case. We aimed to capture genuine interactions and decision-making processes while allowing flexibility for participants to suggest feature improvements that better align the tool's workflow with their ProtocolREing process.

3.3 Recruitment

We posted on reverse engineering-related channels on social media (e.g., LinkedIn and Reddit), posted on Slack channels for various popular reverse engineering tools (e.g., BinaryNinja and Ghidra), shared our recruitment message with the leadership of social organizations focused on reverse engineering, and sent emails to a curated panel of security professionals who made themselves available for user studies. Anyone who responded to these recruitment efforts was asked to fill out a screening survey that asked questions about experience with ProtocolREing. Specifically, we asked participants to rate their skills in reverse engineering, vulnerability discovery, malware analysis, and software development on a Likert scale ranging from "Basic (Limited exposure)" to "Expert

(An acknowledged source of knowledge)." Additionally, we asked respondents to submit their resumes detailing their reverse engineering experiences to contextualize their responses and help us understand their background and familiarity with ProtocolREing.

We interviewed all candidates with self-reported "advanced" or "expert" skill levels in at least two categories. P5 did not provide their experience level in the survey, but from their CV, we determined they met our interview criteria. We also required participants to be English-speaking and at least 18 years old. While our recruitment focused on smaller communities focused on reverse engineering, we still received several inauthentic responses. We removed responses with duplicate answers to open-ended questions and very similar responses (e.g., email addresses with a few letter changes) within a short period. While this slowed recruitment efforts, it did not impact our data collection as it was straightforward to determine whether a participant *actually* had ProtocolREing experience once the interview began. Specifically, one participant was removed from our study and not included in our analysis because they did not understand basic ProtocolREing concepts, such as converting hex to ASCII, and could not provide specifics about their process when asked to elaborate. Additionally, this participant reported conflicting information during their pre-screening survey and at the beginning of the interview (i.e., their job title and roles performed), indicating issues with this participant.

All participants were compensated with an Amazon.com gift card valued at $40 USD.

3.4 Analysis

In this study, we employed iterative open coding [68, pg. 101-122] to analyze the interviews and identify key themes related to participants' approaches to ProtocolREing and their interaction with the automations' suggestions. We primarily followed an inductive approach, allowing codes to arise from the data. However, we used some predefined codes from previous work as a foundation to guide our framing. Specifically, we drew initial codes regarding reverse engineers' processes and usable tool guidelines as described in Section 3.1.2 and from prior studies [48, 75]. We expanded on these as necessary to capture unique aspects of ProtocolREing.

The interviewer and one additional researcher collaboratively coded two interviews and discussed the results with the entire research team to develop the initial codebook. The codebook, given in Appendix A, centered around three high-level categories: their ProtocolREing processes, information needs, and automation interaction needs. We also code other items, such as their doubts about the automation's suggestions and any additional requested features. Because we presented multiple views, we applied codes uniquely for each view to capture differences between their components. This allowed participants to preempt features provided in future views, suggesting that those features are general needs instead of specific needs in certain contexts.

Using this codebook, two researchers independently coded the remaining 14 interviews, meeting after every two interviews to compare codes and resolve disagreements. After each round, the full research team discussed the codes and updated the codebook accordingly; these changes were reapplied to previously coded interviews. Because the researchers jointly reviewed every coded transcript, we did not calculate inter-rater reliability [49].

Next, we performed axial coding to identify connections within and between our applied codes, allowing us to extract higher-level themes and relationships [68, pg. 123-142]. We specifically wanted to identify the different information and interaction needs and our participants' reservations regarding automation.

3.5 Limitations

This research shares limitations that are common to other exploratory qualitative work. First, our sample size is small, and our results may not generalize. For most findings, we indicate the portion of participants who stated the theme to provide a general indication of prevalence. However, participants who do not mention a theme may still agree but may not have thought to mention the theme. Therefore, we chose not to perform statistical comparisons, and our prevalence results may not generalize beyond our sample; instead, we suggest directions for future work. We attempted to recruit broadly to capture diverse perspectives. Because we reach saturation of themes, we expect our results to represent many approaches to and opinions on ProtocolREing.

Our results may also be limited by the specific ProtocolREing scenario we used as an example and the prototype views we began with. This priming likely introduced some anchoring bias, i.e., participants are more likely to focus on information and interactions presented than they might if shown other views first [44]. We attempt to account for this bias by utilizing multiple views and developing views to represent a range of existing tools through expert feedback and our tool review (see Section 3.1). We chose our scenario as it is representative of a common class of ProtocolREing problems, i.e., interacting with embedded and legacy systems, and required participants to consider various data types and relationships between fields. We also included probing questions to encourage participants to think beyond our views and scenario. We believe we achieved this as several elements of our ProtocolRE process diagram in Figure 4 were inspired by our participants even though they fell outside the direct actions they took in our scenario.

Our study was conducted in a lab setting with a paper prototype. When performing ProtocolREing, participants can directly interact with tools and utilize additional resources like running the code with different configurations, reverse engineering associated binaries, or reading device documentation. While our study does not capture their full ProtocolREing process, it does elucidate their interaction needs with automation and their information needs to complete their tasks properly. Furthermore, several participants remarked that the Combined View was familiar and reflected the types of interfaces they encounter in their daily work. Future work is required to investigate other facets of the ProtocolREing process.

4 Ethical Considerations

Before the study, participants were informed about the data collection practices, including the nature of the collected data, how it would be used, and the various measures to protect their confidentiality. Each participant provided informed consent, indicating their voluntary agreement to participate in the study. They were made aware participation was entirely optional and they could withdraw at any time without penalty. Additionally, participants could skip questions or tasks they preferred not to answer. This study was reviewed and approved by our institution's ethics review board.

5 Results

This section presents our findings. We start by describing the participant population before addressing our thematic findings on ProtocolREing process, ProtocolREs' information needs, and ProtocolREs' interaction needs. We refer to participants using the notation "PX" where X is an anonymized number unique to each participant. See Table 2 for more details on each participant. As we are presenting qualitative results, we do not report participant response counts for observed themes as these can be misleading. Instead, we use quantifier terminology this is a qualitative interview study, we do not report exact participant response counts. Instead, we use general quantifier terminology, which is conventional in qualitative research [18] and matches prior similar works [28, 42, 82].

5.1 Participants

Our participants provided diverse demographic and professional backgrounds, as described in Table 2. Of our 16 participants who completed the interview, one identified as non-binary, one declined to share their gender, and the remaining identified as men (N=14). While ethnicity was an optional category, 13 participants self-identified as White. Most reside in the US (N=11), and five were located outside North America, including Eastern and Western Europe. Participants were between 19 and 44 years old, with the average age being 33.8 years old. Participants were highly educated, almost all holding a Bachelor's degree (N=13), and about half (N=7) had graduate degrees. This is consistent with the demographics from prior large-scale surveys of SoftwareREs [10, 29, 30]—there are no similar surveys of ProtocolREs to our knowledge—, but our population skews slightly older and more educated. This is expected as we targeted experts. Participants reported various roles and titles in software development, security research, reverse engineering, and academic roles, such as PhD students with a focus in reverse engineering.

We did not observe any difference in the processes, information needs, or interaction preferences between participants who reported intermediate versus advanced reverse engineering experience; therefore, we do not make direct comparisons between experience levels.

5.2 ProtocolREing Process (RQ1)

Based on participants' interactions with the prototype and the discussion of their ProtocolREing process, we observed their processes could be described as an iterative cycle of hypothesizing parts of the protocol and assessing whether these definitions fit the data. We present the components of this process model in Figure 4. Participants' hypotheses centered on asserting potential field boundaries (Figure 4.C) and datatypes (Figure 4.D), testing how different configurations challenge or support their beliefs. In this section, we discuss how participants interleave identifying and assessing hypotheses across these dimensions, supported by additional external information to produce a final protocol, as shown in Figure 4.G.

Participants used landmarks and/or neighboring bytes as indicators to determine field boundaries (Figure 4.C). All of our participants expressed the importance of identifying field boundaries during ProtocolREing. Participants primarily described two methods for determining field boundaries: 1) Investigating differences between neighboring bytes or fields. 2) Using easily identifiable patterns as "landmarks" to set field boundaries.

When establishing initial potential field boundaries, most participants described using differences in neighboring bytes to determine natural breaks. For example, P8 chose to segment into fields based on differences in the data distributions, saying they chose that field boundary *"because the repeating parts are grouped together, and the non-repeating parts are grouped in a way that makes sense."* ProtocolREs can use these subtle indicators between bytes to set initial field boundaries for further investigation.

Additionally, most participants described landmarking using typical networking values like magic numbers or ASCII characters. When evaluating the automation's suggested field boundaries, P16 described how *"Field one could definitely be like a magic value of some sort. This is the same for every packet, just like a protocol identifier."* These repeating patterns across multiple messages were a clear indicator to determine field boundaries or details about the data sample. The majority of participants described identifying trends across multiple messages, like P9 said in the Decoding View: *"I'm looking at things as columns... and then I see 301 300 298 and I see these values are monotonically decreasing. .... So now I'm thinking the messages are out of order in the original data."*

Participants' assumptions about datatypes in the protocol supported or disproved their hypotheses (Figure 4.D). One technique nearly all participants described was using a common, easily recognizable datatype, typically ASCII, as an anchor point to establish initial field boundaries. Most participants recognized ASCII characters from the raw hex values and used bytes inside the valid ASCII range to set initial field boundaries like P8: *"It looks like some kind of ASCII characters because the ranges of the bytes fall in ASCII range."* Similarly, P2 recognized the hex bytes as valid ASCII characters and wanted the automation to take this into account: " *I would like to be able to do something about that to affirm to the algorithm. That's good. Please take this into account."* This idea of incorporating user feedback is described further in Section 5.4.

Once the automation decides a type to suggest or the ProtocolRE produces a hypothesized data type, almost all participants wanted the data decoded as that type. By decoding the data as the specific type in-line, ProtocolREs can quickly assess whether the type fits. For example, when trying to determine the type of one of the fields, P7 said, *"I think that field four should be ASCII characters, so if I can tell [the automation] that should be represented as ASCII and see it as that immediately, that would be helpful."*

Several participants found the data distribution presented in the Histogram View helpful when determining datatypes. P2 scanned the Histogram View and was able to determine the final field was ASCII because all the values were contained within the ASCII range, saying, *"the obvious signal amongst this noise is the ASCII characters at the end of the messages."* However, several participants felt the Histogram View was only useful for initially providing an overview

ID	Role	Education	RE Experience	Vuln discovery	Malware analysis	Software dev
P1	Developer	Doctorate	Intermediate	Basic	None	Expert
P2	PhD Student	Master's	Advanced	Intermediate	Advanced	Advanced
P3	PhD Student	Bachelor's	Intermediate	Intermediate	Basic	Advanced
P4	Developer	Some college	Advanced	Advanced	Intermediate	Advanced
P5	Reverse Engineer	Bachelor's	Expert	-	-	-
P6	Reverse Engineer	Master's	Expert	Intermediate	Intermediate	Expert
P7	Other (Security)	Bachelor's	Expert	Expert	Intermediate	Advanced
P8	Developer	Bachelor's	Expert	Intermediate	Intermediate	Advanced
P9	Reverse Engineer	Some college	Advanced	Advanced	Intermediate	Advanced
P10	Reverse Engineer	Master's	Intermediate	Beginner	Advanced	Beginner
P11	Developer	Some college	Intermediate	Intermediate	Basic	Expert
P12	Other (Security)	Doctorate	Advanced	Intermediate	Basic	Basic
P13	Leadership	Bachelor's	Expert	Expert	Intermediate	Intermediate
P14	Leadership	Doctorate	Expert	Expert	Expert	Advanced
P15	Other (Security)	Master's	Intermediate	Intermediate	Basic	Intermediate
P16	Reverse Engineer	Bachelor's	Expert	Intermediate	Expert	Intermediate

Table 2: Participant ID, functional roles, education, and reverse engineering skills.

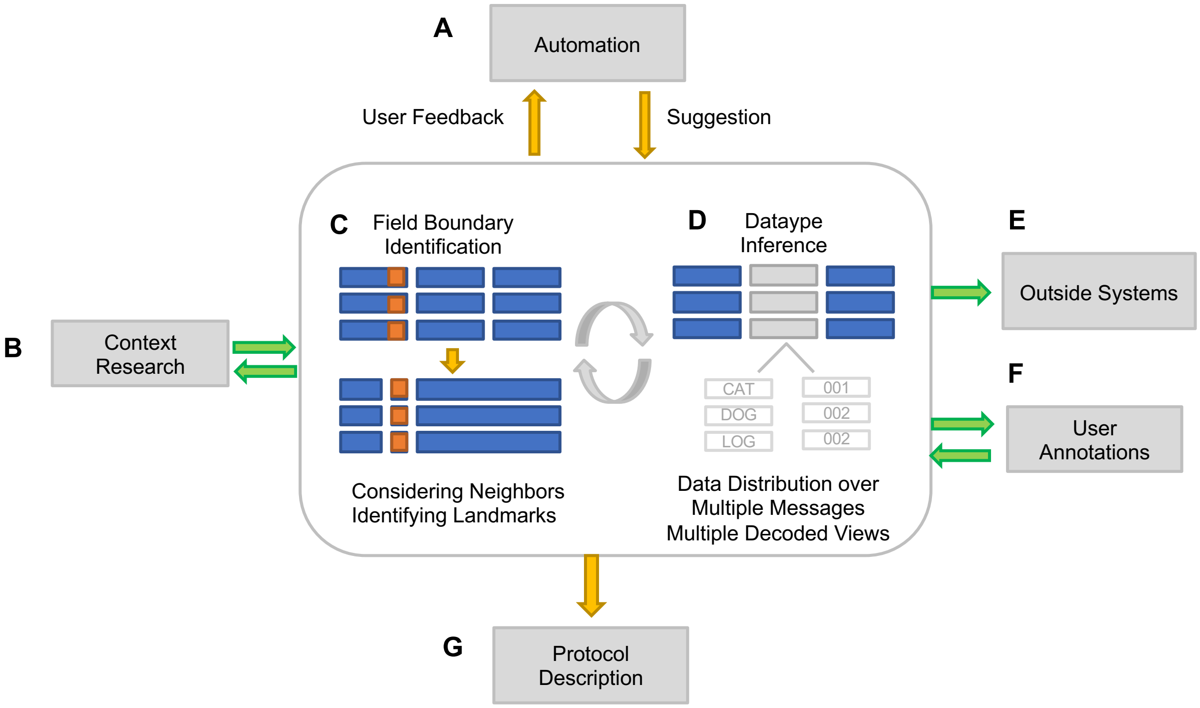


Figure 4: Illustration of the Protocol reverse engineering (PRE) process. The core components, Field Boundary Identification (C) and Datatype Inference (D) interactively drive the development of a Protocol Description (G) describing the data, field boundaries, and datatypes. Automation (A) and Contextual Research (B) each inform and are guided by both the current description and user feedback. Users typically start by performing Contextual Research (B) or by identifying fields (C) and types (D). Outside Systems (E) use the description and data for further validation. Throughout the PRE process, User Annotations (F) allow the user to store and recall ideas about the data and their hypotheses.

of the data but did not provide any additional insights. For example, P15 explained, *" once you realize that it's ASCII text, the histogram is no longer useful."*

Lastly, several participants described how decoding hex values to several datatypes simultaneously benefited their process. P9 described how the tool he typically uses for his job, *"[has] the* *ability to visualize things side by side and simultaneously visualize more [datatype] interpretations at once."*

Participants regularly switched between identifying field boundaries and datatypes (Figure 4.C-D). We found the relationship between field boundaries and datatypes is tightly coupled. ProtocolREs often use their hypotheses about the field boundaries to inform hypotheses about the datatypes, subsequently adjusting their field boundaries and continuously iterating cyclically. Several participants described how a small change to one field can lead to a cascade of other updates. P1 explained, *"For example, you adjusted this boundary now, and the delta in Field 2 became this much more variable across the whole data set also, probably per field."* Because these two aspects are so interconnected, we observed all participants regularly switched between considering datatype and field boundary hypotheses as new patterns and ideas arose.

Participants want automation to integrate with other ProtocolREing systems and information (Figure 4.B,E). Most participants expressed the importance of integrating multiple input and output streams. Early in the interviews, participants often wanted to do context research, like reviewing the binary and source code, comparing findings to established templates, or looking at documentation. P9 said, *"The truth is in the binary. If you can get access to the binary that you can reverse engineer, that is often the quickest way to deal with things."*

Other participants described turning to documentation related to the application they were investigating that might reveal some information about the protocol. For example, P11 said, *"I would go look at the RFC [Request for Comments] … to understand how the protocol is designed."* Similarly, P5 described looking for patents, saying, *"Is there a patent on these people?' so that this way I can find what they think they're doing."*

Several participants described transferring the results of their ProtocolREing efforts to other tools for testing or use in other applications. For example, P15 described their process of switching between multiple tools: *"If this is a lot of traffic, we filter and automate Scapy extraction of data … Then we can change, and can state in a descriptive language like in Kaitai-struct a definition [of the protocol]."* As another example, P14 described wanting the automation to generate scripts usable by other packet processing tools, saying, *"I would like it to generate decoder code that fits the tool that I am using [Wireshark]."*

Participants emphasized the importance of message ordering by avoiding the provided filtering and sorting features. We initially thought filtering and sorting features would be beneficial. However, participants had mixed reactions. Of the ten participants who interacted with these features, half felt they were not helpful, as filtering and sorting could disrupt the sequence and ordering of messages. P16 explained, *"Preserving the time of the messages and showing them in time order. It's extremely important."*

However, a few participants liked the idea of filters as they would help focus the view only on relevant messages. For example, P9 wanted to utilize filtering visually in the Histogram View to focus on messages within a certain value range, saying *"I'd want to be able to drag a region in the visualization and have it filter the rest of the data dynamically to only that data. It lets you immediately see that the field one values never change; that the field two values are somewhat periodic, and that the field three values never really change."*

5.3 Information Needs From Automation (RQ2)

Given participants' ProtocolREing process from the prior section, we now discuss themes in the types of additional information participants wanted when working with automation. In particular, we observed two general themes. First, participants often did not expect the automation to be exactly correct due to the ambiguity of the task; therefore, they wanted the automation to take a conservative approach and provide mechanisms for evaluating the outputs. Also, participants wanted ways to inspect and compare multiple suggestions by the automation.

5.3.1 ProtocolREs do not trust automation to provide a complete solution, but see value in automation as a support agent. When considering the suggestions, most participants doubted that any automation could be sufficiently accurate to correctly determine a full protocol due to challenges inherent to ProtocolREing. That is, finding samples to train or test ProtocolREing automation on, especially for malware analysis, is difficult due to the rapidly changing and adversarial nature of malware [52, 69]. Several participants shared a similar sentiment to P16, who said, *" Most of the time we're dealing with new and novel stuff [that] isn't in the [training] data set."* Other participants believe current automation does not scale to complex real-world examples. For example, P14 said, *"They don't work on real software. That's why I say automated reverse engineering tools don't really exist because they only really exist on paper."*

Despite participants' belief that automation would make mistakes when automatically identifying a protocol, most participants believed the suggestions were still helpful as a starting point for their analysis and to give them ideas for further analysis as an interactive support agent. For example, P12 said, *"the more options that the system gives and says, Hi, we think it is this… the more that the system says here is a possible suggestion the better."*

Prior work investigating trust in other uses of automation has shown users initially distrust automation, though this reduces with use as they learn its boundaries [41, 60]. Several participants expressed a similar sentiment to P10 when using automation, *"If I'm using the tool and find that the suggestions often are helpful, Then over time my trust in it would develop."* Similarly, P16 described learning an automated tool's *"individual quirks, and if it's suggestions are going to be helpful or not, and what kinds of things it's better it's suggesting."*

Automation should make conservative field boundary and datatype edits. Most participants wanted automation only to edit the presented fields and data types when it was confident and present information with a minimal visual footprint otherwise. P16 said liberal field boundary and datatype edits place a burden on ProtocolREs, creating extra work, saying *"it's just like 15 other things that you have to go through and like, minimize and click through."* This is particularly problematic if users cannot override suggestions or limit the scope of changes made by the automation. For example, P8 described how this would be frustrating, saying, *"If I assign one* *type, and then it goes and re-assigns types for other fields that I have already reviewed the suggestions for, it would mess me up."*

Participants want semantic characteristics about the data as suggestion explanations. In addition to wanting the automation to provide conservative suggestions, most participants wanted ways to assess and understand those suggestions, then make refinements (which we discuss further in Section 5.4). P16 said, *"With automated tools like this, you kind of have to get some experience with them to figure out their individual quirks. And if it's suggestions are going to be helpful or not, and what kinds of things it's better at suggesting."* To help them make these assessments, participants wanted the automation to justify its suggestions. P1 said, *"I'd want to understand what information is going into that prediction. Why is it picking that?"*

Providing these explanations for automation results is a common request beyond ProtocolREing [27, 53]. However, we further refine this need by observing ProtocolREs wanted these explanations presented as characteristics of the data fitting the suggested datatype's range. For example, P15 wanted to know if a datatype suggestion was based on a relationship between two fields, explaining the automation should show *"the length of this field is controlled by this field over here."* Similarly, P9 described a checksum relationship between two field, saying *"these 2 bytes of this data might be a checksum because they are a valid CRC16 with X,Y,Z parameters [fields]."* Additionally, many described looking for these datatype characteristics when inferring the datatypes manually. For example, most participants investigated whether a field's data was restricted to the ASCII range, indicating it was an ASCII string, or monotonically increasing, likely a counter.

Several participants believed adding some indicator of the automation's confidence in a result, based on these characteristics, would help users make assessments. P13 described this, saying, *"For any automated tool, I like to show confidence level. [What] does the tool really think percentage wise? This is 90%, it's probably certain, is it 85? We're not really confident, but we have a good guess. Is it 50? We're just guessing out of the blue... That helps not having to go down a wild goose chase."*

5.3.2 ProtocolREs perform analysis by comparing data both within single messages and across multiple messages. Due to the complex nature of ProtocolREing and the large number of analyzed messages, participants appreciated features that allowed different forms of comparison. The Combined View included allowing quick comparisons across all messages using the histogram, viewing multiple messages simultaneously, and seeing the data decoded into multiple datatypes simultaneously. P9 described their information needs saying, *"workflow-wise... [what] I'm used to is much more parallel... viewing the data in multiple ways at the same time."*

However, excess visual information can overload ProtocolREs; thus, several participants described wanting to hide parts of views once they were no longer needed. Cluttered visuals are a common challenge in interface design, but our results indicate when different information should be presented. For example, P16 felt the histogram was not needed at all times: *" Having the ability to hide them is useful because it's something I'll look at once, and then I never look at them again."* Similarly, P13 described wanting to *"just 'x' [the float and integer windows] out, and they would leave and then say the Hex data or the ASCII field would automatically resize bigger."* While participants discussed wanting to hide many of the views, no participant indicated they wanted to hide the Split-Merge View, likely because this interaction with fields and data types is central to ProtocolREs' processes.

Participants mostly wanted to view each field boundary decoded according to its suggested datatype, not multiple datatypes. To assess the validity of a suggestion, participants often focused on the datatype representation. This included wanting to see the data decoded as the specific datatypes on a per-field basis. For example, P6 said, *"If it could break the message, some of it being ASCII, some of it being the floats, some of it being an integer while still displaying each individual message broken up as a comparison, that would be more helpful."* This allows ProtocolREs to visualize the parsed protocol data and assess whether the presented data matches the ranges they would expect to see. However, presenting the message decoded as each datatype simultaneously, as we did (See Figure 2.D), was viewed as overload for many participants. Instead, they preferred some drop-down option, allowing quick switching between decoding types for comparison instead of showing all at once. P2 explained, *"There is totally enough mental bandwidth to say, 'click', and then within a millisecond field four changes [to ASCII]. And then I could just click back to integer if it doesn't help."*

Participants wanted to know how datatype suggestions changed when field boundaries were moved. As the automation provides different field boundary suggestions or the user splits or merges fields, this would cause the automation to suggest additional datatypes for newly formed fields. Some participants indicated they wanted a way to see each of these states of the field boundaries and compare how changes to the fields impact within-field variation and patterns. For example, P1 said, *"I want it to say, for example, you adjusted this boundary, now field 2 became this much more variable."* This follows from the fact that we observed ProtocolREs regularly switching between assessing field boundaries and data types; thus, information helping them see aspects of this relationship was beneficial.

5.4 Interaction Needs With Automation (RQ3)

Finally, we discuss how participants wanted to interact with the automation. These include programmatically extending datatype descriptions, interacting with suggestions and reverting actions, and automatically generating scripts for future ProtocolREing process steps.

Participants wanted extendable datatype descriptions that consider multiple fields. In the Prototype Interface, we only showed a few basic datatypes, but participants discussed wanting the automation to identify a wide range of possible complex non-traditional datatypes like sequence counters or message identifiers. P14 wanted the automation to recognize non-traditional datatypes like hashes saying, *"It would save me time identifying what things are probably integrity checks like series or Md5 or check digits."* Because some semantic types might be specific to the studied application, participants wanted the ability to write their own type definitions or provide something like C struct definition that could help the automation recognize these relationships in the data. This would also support alternative data encoding—a need expressed by almost all participants—, such as specific string encodings. P9 explained, *"I'd feel limited by ASCII. I know this is probably ASCII data, but what if it's not a normal code page? what if it's CJK encoding, which is common in China, or Shift-JIS encoding, which is common in Japan?"*

As discussed in Section 5.2, many participants attempted to determine datatypes by looking at relationships between fields, such as message type identifiers or checksums. A few participants reported these trends are typically harder for ProtocolREs to identify by hand; thus, they wanted additional feature support to help identify these relationships, which could be supported through extended datatype representations.

ProtocolREs want to guide automation by constraining field boundaries and datatypes. As discussed previously, most participants expect the automation to make some mistakes due to the problem's complexity. Therefore, most participants wanted to incrementally guide the automation by changing a field's datatype or boundaries and forcing the automation to use those changes when determining future suggestions. P13 explained, *"Most automated testing a lot of times is incorrect…I know patterns because that's something that I use all the time. Automated systems don't always know these patterns, or they don't know something that I've seen before, but they haven't. And so, having that flexibility to change it and then see, does that correct it if I think something is wrong, or does that make it perform better?"*

ProtocolREs need the ability to switch back and forth between possible automation suggestions quickly. As we discussed in Section 5.2, ProtocolREs regularly switch back and forth between field boundaries and datatypes, testing whether different possible protocol specifications fit their data. Therefore, when interacting with the automation, several participants emphasized the importance of quickly and easily reverting actions or configuration settings while working. For instance, they could test how changes in field boundaries or datatypes might impact their understanding of the protocol. Then, they can quickly revert or compare these changes to their original analysis. P7 described using a preview of changes to make a comparison, saying *"if [there's] some way to split [fields] up and then undo it…preview how it would affect the guesses or the field values, so you split it at this point and then you can see the values of the [fields] and what the guess type for the new fields would look like."*

Similarly, several participants wanted ways to maintain their mental models and understanding of the data. This can take numerous forms, from leaving in-line comments or sticky notes as P5 described: *"If I'm looking at this in three weeks, when I get pulled to something else; If I come back I have no [idea] when I figured out what this ASCII meant. I have no idea what it meant to me back then."*

Participants wanted support for generating data to validate automated suggestions and their protocol inferences. Many participants wanted a mechanism to provide additional input messages or generate output messages that could be validated with an external tool (Figure 4.E). Some participants wanted to manually craft new messages to see how specific changes would impact the automated suggestions. For example, P2 said, *"I would like to be able to add a new message and arbitrarily manipulated bytes, and see in real-time how it gets parsed as a float. That would easily allow me to test hypotheses about what this program is showing me."*

Alternatively, half of the participants requested some form of script generation based on their inferred protocol definitions to use with other tools to assess the validity of their final protocol definition. For example, Wireshark and other packet parsing tools allow users to provide data decoders for new protocols, using these decoders to parse received packets. Multiple participants mentioned they wanted the automation to generate a decoder that could be applied to a packet parser and give it a large volume of packets. If the parser fails, this would indicate the protocol definition is incorrect. For example, P14 said, *"I'd want [the prototype] to be provable…I probably want it to be something that emits C code that then can handle decoding whatever packet capture you like, or maybe it emits a Wireshark dissector."* Other participants proposed validating their inferred protocols by automatically producing code that would generate packets matching the inferred protocol. Then, these packets could be sent to a server that uses the studied protocol to see if it responds as expected. As P14 explained, *"I would generate packets and send them at real servers and see what happens. The live poking of things can be really useful as a lot of them will drop packets on the floor if they're wrong or respond with an error."* A few participants pointed out these scripts would also provide a dual use, as they could be directly used for future security tasks like intrusion detection systems.

6 Discussion

Based on the interviews with professional ProtocolREs, we observed the process model described in Figure 4, where ProtocolREs cyclically iterate between adjusting field boundaries (Figure 4.C) and establishing datatypes for fields (Figure 4.D). These decisions are tightly coupled, and ProtocolREs incorporate information from various external sources, including context research about the application (Figure 4.B) and automated suggestions (Figure 4.A). Participants investigated multiple hypotheses in parallel and preferred to view data in various formats simultaneously. Participants expected the automation would make some errors, especially for novel data; thus, many participants wanted the automation to take a conservative approach, provide additional information related to the data's semantic characteristics to assess suggestion accuracy. They wanted automation interaction through user-defined custom datatypes (Figure 4.F) and allowing the user to switch back and forth between suggested protocol definitions and constrain user-specified field boundaries and datatypes (Figure 4.A). Lastly, participants valued the ability to interface with other ProtocolREing tools, like Wireshark, to validate their inferred protocols (Figure 4.E).

ProtocolREing differs from SoftwareREing in the distinction between generating and testing hypotheses. While we expected there would be some similarity between ProtocolREing and SoftwareREing as they have similar goals and are often performed by the same people, there are key differences in process due to the inherent difference in the ambiguity of the underlying task, as discussed in Section 2.1. Comparing our process model (Figure 4) with three-phase SoftwareREing model defined in prior work [9, 76], they are both hypothesis-driven and iterative. However, the clear difference is that the ProtocolREing process is more condensed, meaning there are not clear distinctions between parts of the process as participants discuss performing hypothesis generation and testing using the same interfaces and sometimes simultaneously. Thus, we present a cyclic process centered on the dual problems of field and data-type identification, whereas prior work found the SoftwareREing model can be broken into three distinct phases SoftwareREs progress through and use different tools to support, dividing hypothesis generation and testing. This distinction creates unique needs for ProtocolREing automation and interaction, as ProtocolREs want to use automation as an interactive support agent, and cannot trust it to produce final answers.

6.1 Recommendations for ProtocolREing Automation Developers

As we discussed in Section 1, while ProtocolREing differs from SoftwareREing due to the ambiguity of the analysis, it is similar to problems in data science where analysts visualize and consider data records in different ways to reveal relationships. We saw this play out with the information and interaction needs identified (Sections 5.3 and 5.4, respectively) as participants wanted to see the data presented in many different ways, wanted the automation to provide explanations for suggestions, wanted the ability to validate and correct the automation, and wanted to get this information without creating information overload. These are common challenges when using automation to support visual data exploration tasks, thus these similarities suggest existing guidelines from this domain [20, 31, 58, 73, 81] could be leveraged for future automation design. However, our results provide additional insights regarding how these can be best explicitly applied to support ProtocolREs. Specifically, we observed two overarching themes for interface design based on participants' responses. We discuss each in turn.

Organize interactions around datatypes and field boundary characteristics. Throughout our interviews, we observed participants' investigations centered around the unique semantic characteristics of different datatypes and restrictions on possible field boundaries. These components were central to their processes, and shaped their interaction needs. For example, participants specifically liked the Histogram Viewbecause it allowed them to determine whether data in a field matched the known bounds of particular datatypes. However, because the view did not allow them to view inter-field relationships, they wanted it extended to capture richer datatype representations. Additionally, the only information we did not hear anyone say they wanted to hide was the field boundaries and datatype labels, further indicating the centrality of this information. Therefore, any ProtocolREing interaction should be designed around datatypes and field boundaries.

Focus interfaces on supporting iterative investigation. The second theme arises from the fact that our participants expected automation to make erroneous suggestions but still saw these tools as helpful in supporting their process of quickly testing variations on potential protocols. They quickly move through the search space, swapping a datatype or field boundary, and want automation to take a similar approach, conservatively suggesting changes to parts of the protocol definition when confident. To support this iterative search, automation should enable easy comparison by neighbor points in the search space by allowing analysts to preview or undo a suggestion and visualize the impact of a change.

6.2 Recommendations for Researchers

While this study provides direct recommendations for tool development, we can draw out valuable insights for researchers investigating ProtocolREing. First, we identify a process model for ProtocolREing, which can be used to guide the development of controlled empirical studies of the various tasks. Our study was conducted in an observed setting without a working tool, but future work should build on our foundation to test with a live tool and networking data. This would allow future research to answer questions about the most effective common practices and strategies and what interfaces help save time and streamline the process for ProtocolREs. Similarly, we enumerated the set of ProtocolREs' information and interaction needs, but our study design prevents us from identifying which would be the most useful. Developers must make tradeoffs with limited development time, especially in this domain where many tools and plugins are produced by the open source community in their own time [48], so assessing actual feature value is necessary.

The information and interaction needs we identified should be assessed with additional populations and settings. First, because we only test with expert ProtocolREs, whether these requirements hold for beginners is not apparent. Repeating our study directly with beginners will likely not be sufficient as beginners likely lack the experience to envision what they might be missing. However, given the additional information and interactions, a future study could assess whether these are helpful for beginners. This would be most useful in defining initial universally usable default views. Finally, future work should investigate how these processes change in alternate settings, specifically when users can access application source code or binary files. Work in this setting will provide a bridge between our work and the prior SoftwareREing literature [6, 9, 11, 24, 46, 75, 77], likely unearthing unique insights at their intersection.

7 Conclusion

While some tools automate protocol RE, none have a user interface for ProtocolREs to guide the automation. Therefore, we sought to understand how best to build an interface for this task to support ProtocolREs. To this end, we interviewed 16 ProtocolREs, asking them to interact with a paper prototype and provide feedback. We found ProtocolREs ground their process by investigating two core components of network protocols, the field boundaries, and the datatypes. These central actions are supported by numerous information streams, including automated tooling support, context research, user annotations, and feedback. Further, we found they expected any automation to make mistakes and to be unable to describe a full protocol due to ambiguity. Therefore, human-automation interaction work is particularly necessary so ProtocolREs can actually use and incorporate automation into their workflow.

Finally, we found these interactions should be tailored to ProtocolREing by focusing on the datatype characteristics and relationships between field boundaries, which provide the underlying structure ProtocolREs rely on to search the possible protocol space. These interactions should focus additionally on the rapid exploration of the search space by comparing different field and boundary possibilities and allowing ProtocolREs to apply and undo automation suggestions and investigate numerous hypotheses simultaneously. Our results will help the development of tooling interactions necessary to make ProtocolREing automation more usable and widely adopted in the field.

Acknowledgments

Many thanks to the anonymous reviewers who provided helpful comments on drafts of this paper and Jordan Wiens for help with recruitment. This project was supported by NSF grant CNS-2247954. The first author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author. MITRE has approved this work for public release and unlimited distribution; public release case number 24-3822.

References

[1] Kulsoom Abdullah, Christopher P Lee, Gregory J Conti, John A Copeland, and John T Stasko. 2005. IDS RainStorm: Visualizing IDS Alarms.. In *VizSEC*. 1.

[2] Miltiadis Allamanis, Earl T. Barr, Christian Bird, and Charles Sutton. 2015. Suggesting Accurate Method and Class Names. In *2015 Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE '15)*. Association for Computing Machinery, New York, NY, USA, 38–49. doi:10.1145/2786805.2786849

[3] AllsafeCyberSecurity. 2021. Awesome Ghidra. https://github.com/AllsafeCyberSecurity/awesome-ghidra. (Accessed 08-11-2021).

[4] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2019. Code2vec: Learning Distributed Representations of Code. *Proc. ACM Program. Lang.* 3, POPL, Article 40 (Jan. 2019), 29 pages. doi:10.1145/3290353

[5] Dejan Baca, Bengt Carlsson, Kai Petersen, and Lars Lundberg. 2013. Improving software security with static automated code analysis in an industry setting. *Software: Practice and Experience* 43, 3 (2013), 259–279. http://dblp.uni-trier.de/db/journals/spe/spe43.html#BacaCPL13

[6] Steffen Becker, Carina Wiesen, Nils Albartus, Nikol Rummel, and Christof Paar. 2020. An Exploratory Study of Hardware Reverse Engineering — Technical and Cognitive Processes. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, 285–300. https://www.usenix.org/conference/soups2020/presentation/becker

[7] Benjamin Bichsel, Veselin Raychev, Petar Tsankov, and Martin Vechev. 2016. Statistical Deobfuscation of Android Applications. In *2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 343–355. doi:10.1145/2976749.2978422

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[9] Adam Bryant. 2012. *Understanding How Reverse Engineers Make Sense of Programs from Assembly Language Representations*. Ph. D. Dissertation. US Air Force Institute of Technology.

[10] Bugcrowd. 2023. Inside the Mind of a Hacker. https://www.bugcrowd.com/wp-content/uploads/2023/11/Inside-the-Mind-of-a-Hacker.pdf. Accessed: 2024-08-17.

[11] Kevin Burk, Fabio Pagani, Christopher Kruegel, and Giovanni Vigna. 2022. Decomperson: How Humans Decompile and What We Can Learn From It. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 2765–2782. https://www.usenix.org/conference/usenixsecurity22/presentation/burk

[12] Janis A Cannon-Bowers and Eduardo Ed Salas. 1998. *Making decisions under stress: Implications for individual and team training.* American psychological association.

[13] Jared Chandler, Adam Wick, and Kathleen Fisher. 2023. BinaryInferno: A Semantic-Driven Approach to Field Inference for Binary Message Formats.. In *NDSS*.

[14] Daniel Collins. 2024. ReHex. https://github.com/solemnwarning/rehex/. (Accessed 08-11-2024).

[15] Greg Conti. 2007. *Security data visualization: graphical techniques for network analysis*. No Starch Press.

[16] Gregory Conti and Kulsoom Abdullah. 2004. Passive visual fingerprinting of network attack tools. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*. 45–54.

[17] Gregory Conti, Erik Dean, Matthew Sinda, and Benjamin Sangster. 2008. Visual reverse engineering of binary and data files. In *International Workshop on Visualization for Computer Security*. Springer, 1–17.

[18] Shelby Corley. 2024. How to "Quantify" Qualitative Data. https://www.evalacademy.com/articles/how-to-quantify-qualitative-data.

[19] DARPA. 2016. DARPA | Cyber Grand Challenge. http://archive.darpa.mil/cybergrandchallenge/ (Accessed 02-26-2017).

[20] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. 2014. Explore-by-example: An Automatic Query Steering Framework for Interactive Data Exploration. In *SIGMOD '14* (Snowbird, Utah, USA). ACM, New York, NY, USA, 517–528. doi:10.1145/2588555.2610523

[21] Adam Doupé, Marco Cova, and Giovanni Vigna. 2010. Why Johnny Can'T Pentest: An Analysis of Black-box Web Vulnerability Scanners. In *Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (Bonn, Germany) *(DIMVA'10)*. Springer-Verlag, Berlin, Heidelberg, 111–131. http://dl.acm.org/citation.cfm?id=1884848.1884858

[22] Julien Duchêne, Colas Le Guernic, Eric Alata, Vincent Nicomette, and Mohamed Kaâniche. 2018. State of the art of network protocol reverse engineering tools. *Journal of Computer Virology and Hacking Techniques* 14 (2018), 53–68.

[23] Daniel Fallman. 2003. Design-oriented human-computer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 225–232.

[24] Irina Ford, Ananta Soneji, Faris Bugra Kokulu, Jayakrishna Vadayath, Zion Leonahenahe Basque, Gaurav Vipat, Adam Doupé, Ruoyu Wang, Gail-Joon Ahn, Tiffany Bao, and Yan Shoshitaishvili. 2024. "Watching over the shoulder of a professional": Why Hackers Make Mistakes and How They Fix Them. In *Proceedings of the IEEE Symposium on Security and Privacy*.

[25] Wireshark Foundation. 2024. Wireshark, the world's most popular network protocol analyzer. https://www.wireshark.org/. (Accessed 08-11-2024).

[26] gchq. 2024. CyberChef. https://github.com/gchq/CyberChef. (Accessed 08-11-2024).

[27] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. LEMNA: Explaining Deep Learning based Security Applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 364–379. doi:10.1145/3243734.3243792

[28] Hana Habib, Sarah Pearman, Jiamin Wang, Yixin Zou, Alessandro Acquisti, Lorrie Faith Cranor, Norman Sadeh, and Florian Schaub. 2020. "It's a scavenger hunt": Usability of Websites' Opt-Out and Data Deletion Choices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*.

[29] Hackerone. 2019. *2019 Hacker-Powered Security Report*. Technical Report. Hackerone, San Francisco, California. https://www.hackerone.com/resources/reporting/the-hacker-powered-security-report-2019

[30] HackerOne. 2020. *The 2020 Hacker Report*. Technical Report. HackerOne, San Francisco, California.

[31] Jeffrey Heer and Ben Shneiderman. 2012. Interactive Dynamics for Visual Analysis. *Commun. ACM* 55, 4 (April 2012), 45–54. doi:10.1145/2133806.2133821

[32] Hex-Rays. 2019. Plug-in Contest 2018: Hall of Fame. https://www.hex-rays.com/contests/2018/index.shtml. (Accessed 05-30-2019).

[33] Hexinator. 2024. Hexinator. https://hexinator.com/. (Accessed 08-11-2024).

[34] ImHex. 2024. ImHex. https://imhex.werwolv.net/. (Accessed 08-11-2024).

[35] Alan Jaffe, Jeremy Lacomis, Edward J. Schwartz, Claire Le Goues, and Bogdan Vasilescu. 2018. Meaningful Variable Names for Decompiled Code: A Machine Translation Approach. In *2018 Conference on Program Comprehension* (Gothenburg, Sweden) *(ICPC '18)*. Association for Computing Machinery, New York, NY, USA, 20–30. doi:10.1145/3196321.3196330

[36] Kaitai-Project. 2015. Kaitai. https://kaitai.io/. (Accessed 08-11-2021).

[37] Samantha Katcher, James Mattei, Jared Chandler, and Daniel Votipka. 2024. Supplementary Material - An Investigation of Interaction and Information Needs for Protocol Reverse Engineering Automation. https://osf.io/ernux/.

[38] Stephan Kleber, Henning Kopp, and Frank Kargl. 2018. NEMESYS: Network Message Syntax Reverse Engineering by Analysis of the Intrinsic Structure of Individual Messages. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*. USENIX Association, Baltimore, MD. https://www.usenix.org/conference/woot18/presentation/kleber

[39] Stephan Kleber, Lisa Maile, and Frank Kargl. 2018. Survey of protocol reverse engineering algorithms: Decomposition of tools for static traffic analysis. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 526–561.

[40] Gary A Klein. 1993. A recognition-primed decision (RPD) model of rapid decision making. *Decision making in action: Models and methods* 5, 4 (1993), 138–147.

[41] Jan H. Klemmer, Stefan Albert Horstmann, Nikhil Patnaik, Cordelia Ludden, Cordell Burton, Carson Powers, Fabio Massacci, Akond Rahman, Daniel Votipka, Heather Richter Lipford, Awais Rashid, Alena Naiakshina, and Sascha Fahl. 2024. Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (Salt Lake City, UT, USA) *(CCS '24)*. Association for Computing Machinery, New York, NY, USA, 2726–2740. doi:10.1145/3658644.3690283

[42] Sabrina Klivan, Sandra Höltervennhoff, Rebecca Panskus, Karoly Marky, and Sascha Fahl. 2024. Everyone for Themselves? A Qualitative Study about Individual Security Setups of Open Source Software Contributors. In *45th IEEE Symposium on Security and Privacy (SP)*.

[43] Jeremy Lacomis, Pengcheng Yin, Edward Schwartz, Miltiadis Allamanis, Claire Le Goues, Graham Neubig, and Bogdan Vasilescu. 2019. DIRE: A Neural Approach to Decompiled Identifier Naming. In *2019 IEEE/ACM International Conference on Automated Software Engineering (ASE '19)*. 628–639. doi:10.1109/ASE.2019.00064

[44] Falk Lieder, Thomas L Griffiths, Quentin J M. Huys, and Noah D Goodman. 2018. The anchoring bias reflects rational use of cognitive resources. *Psychonomic bulletin & review* 25 (2018), 322–349.

[45] Xin Luo, Dan Chen, Yongjun Wang, and Peidai Xie. 2019. A type-aware approach to message clustering for protocol reverse engineering. *Sensors* 19, 3 (2019), 716.

[46] Alessandro Mantovani, Simone Aonzo, Yanick Fratantonio, and Davide Balzarotti. 2022. RE-Mind: a First Look Inside the Mind of a Reverse Engineer. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 2727–2745. https://www.usenix.org/conference/usenixsecurity22/presentation/mantovani

[47] James Mattei, Madeline McLaughlin, Samantha Katcher, and Daniel Votipka. 2022. A Qualitative Evaluation of Reverse Engineering Tool Usability. In *ACSAC 2022, Annual Computer Security Applications Conference.*

[48] James Mattei, Madeline McLaughlin, Samantha Katcher, and Daniel Votipka. 2022. A Qualitative Evaluation of Reverse Engineering Tool Usability. In *Proceedings of the 2022 Annual Computer Security Applications Conference* (Austin, Texas, USA) *(ACSAC '22)*. Association for Computing Machinery, New York, NY, USA.

[49] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and inter-rater reliability in qualitative research: Norms and guidelines for CSCW and HCI practice. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–23.

[50] Gary McGraw and John Steven. 2011. Software [In]security: Comparing Apples, Oranges, and Aardvarks (or, All Static Analysis Tools Are Not Created Equal). http://www.informit.com/articles/article.aspx?p=1680863. (Accessed 02-26-2017).

[51] Matthew B Miles and A Michael Huberman. 1994. *Qualitative data analysis: An expanded sourcebook*. sage.

[52] Jaron Mink, Hadjer Benkraouda, Limin Yang, Arridhana Ciptadi, Ali Ahmadzadeh, Daniel Votipka, and Gang Wang. 2023. Everybody's got ML, tell me what else you have: Practitioners' perception of ML-based security tools and explanations. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2068–2085.

[53] Azqa Nadeem, Daniël Vos, Clinton Cao, Luca Pajola, Simon Dieck, Robert Baumgartner, and Sicco Verwer. 2023. SoK: Explainable Machine Learning for Computer Security Applications. arXiv:2208.10605 [cs.CR] https://arxiv.org/abs/2208.10605

[54] John Narayan, Sandeep K Shukla, and T Charles Clancy. 2015. A survey of automatic protocol reverse engineering tools. *ACM Computing Surveys (CSUR)* 48, 3 (2015), 1–26.

[55] Netzob. 2024. Netzob. https://github.com/netzob/netzob?tab=readme-ov-file. (Accessed 08-11-2024).

[56] Don Norman. 2013. *The design of everyday things: Revised and expanded edition.* Basic books.

[57] Timothy Nosco, Jared Ziegler, Zechariah Clark, Davy Marrero, Todd Finkler, Andrew Barbarello, and W. Michael Petullo. 2020. The Industrial Age of Hacking. In *2020 USENIX Security Symposium (USENIX Security '20)*. USENIX Association, 1129–1146. https://www.usenix.org/conference/usenixsecurity20/presentation/nosco

[58] Adam Perer and Ben Shneiderman. 2008. Systematic Yet Flexible Discovery: Guiding Domain Experts Through Exploratory Data Analysis. In *IUI '08* (Gran Canaria, Spain). ACM, New York, NY, USA, 109–118. doi:10.1145/1378773.1378788

[59] Stephan Plöger, Mischa Meier, and Matthew Smith. 2021. A Qualitative Usability Evaluation of the Clang Static Analyzer and libFuzzer with CS Students and CTF Players. In *2021 Symposium on Usable Privacy and Security (SOUPS '21)*. USENIX Association, 553–572.

[60] Neele Roch, Hannah Sievers, Lorin Schöni, and Verena Zimmermann. 2024. Navigating Autonomy: Unveiling Security Experts' Perspectives on Augmented Intelligence in Cybersecurity. In *Twentieth Symposium on Usable Privacy and Security (SOUPS 2024)*. USENIX Association, Philadelphia, PA, 41–60. https://www.usenix.org/conference/soups2024/presentation/roch

[61] Ben Shneiderman. 1987. *Designing The User Interface: Strategies for Effective Human-Computer Interaction, 4/e (New Edition).* Pearson Education India.

[62] Ben Shneiderman. 2003. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*. Elsevier, 364–371.

[63] Yan Shoshitaishvili, Michael Weissbacher, Lukas Dresel, Christopher Salls, Ruoyu Wang, Christopher Kruegel, and Giovanni Vigna. 2017. Rise of the HaCRS: Augmenting Autonomous Cyber Reasoning Systems with Human Assistance. In *Proc. of the 24th ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM.

[64] Carolyn Snyder. 2003. *Paper prototyping: The fast and easy way to design and refine user interfaces.* Morgan Kaufmann.

[65] Nemesys Software. 2024. Nemesys. https://www.nemesys-soft.com/. (Accessed 08-11-2024).

[66] Sweetscape Software. 2024. 010 Editor. https://www.sweetscape.com/010editor/. (Accessed 08-11-2024).

[67] Robert R Sokal and Charles D Michener. 1958. A statistical method for evaluating systematic relationships. (1958).

[68] Anselm Strauss and Juliet Corbin. 1990. *Basics of qualitative research*. Vol. 15. Newbury Park, CA: Sage.

[69] Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitras. 2022. Expected Exploitability: Predicting the Development of Functional Vulnerability Exploits. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 377–394. https://www.usenix.org/conference/usenixsecurity22/presentation/suciu

[70] Larry Suto. 2010. *Analyzing the Accuracy and Time Costs of Web Application Security Scanners*. Technical Report. BeyondTrust, Inc.

[71] Synalysis. 2024. Synalysis. https://synalysis.com/. (Accessed 08-11-2024).

[72] Jenifer Tidwell. 2010. *Designing interfaces: Patterns for effective interaction design.* " O'Reilly Media, Inc.".

[73] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. SeeDB: Efficient Data-driven Visualization Recommendations to Support Visual Analytics. *VLDB Endowment* 8, 13 (Sept. 2015), 2182–2193. doi:10.14778/2831360.2831371

[74] Vector35. 2019. Vector35/Community-Plugins. https://github.com/Vector35/community-plugins/tree/master/plugins. (Accessed 05-30-2019).

[75] Daniel Votipka, Seth Rabin, Kristopher Micinski, Jeffrey S. Foster, and Michelle L. Mazurek. 2020. An Observational Investigation of Reverse Engineers' Processes. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1875–1892. https://www.usenix.org/conference/usenixsecurity20/presentation/votipka-observational

[76] Daniel Votipka, Seth Rabin, Kristopher Micinski, Jeffrey S. Foster, and Michelle L. Mazurek. 2020. An Observational Investigation of Reverse Engineers' Processes. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/usenixsecurity20/presentation/votipka-observational

[77] Daniel Votipka, Rock Stevens, Elissa M Redmiles, Jeremy Hu, and Michelle L Mazurek. 2018. Hackers vs. Testers: A Comparison of Software Vulnerability Discovery Processes. In *Proceedings of the 39th IEEE Symposium on Security and Privacy (IEEE S&P '18)*.

[78] K. Yakdan, S. Dechand, E. Gerhards-Padilla, and M. Smith. 2016. Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study. In *IEEE S&P '16*. 158–177. doi:10.1109/SP.2016.18

[79] Khaled Yakdan, Sergej Dechand, Elmar Gerhards-Padilla, and Matthew Smith. 2016. Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study. *2016 IEEE Symposium on Security and Privacy (SP)* 00 (2016), 158–177. doi:doi.ieeecomputersociety.org/10.1109/SP.2016.18

[80] Yapeng Ye, Zhuo Zhang, Fei Wang, Xiangyu Zhang, and Dongyan Xu. 2021. NetPlier: Probabilistic Network Protocol Reverse Engineering from Message Traces.

[81] J. S. Yi, Y. a. Kang, and J. Stasko. 2007. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (Nov 2007), 1224–1231. doi:10.1109/TVCG.2007.70515

[82] Shikun Zhang, Yuanyuan Feng, Yaxing Yao, Lorrie Faith Cranor, and Norman Sadeh. 2022. How usable are ios app privacy labels? *Proceedings on Privacy Enhancing Technologies* (2022).

[83] Tianyi Zhang, Zhiyang Chen, Yuanli Zhu, Priyan Vaithilingam, Xinyu Wang, and Elena L. Glassman. 2021. *Interpretable Program Synthesis*. Association for Computing Machinery, New York, NY, USA.

[84] Tianyi Zhang, London Lowmanstone, Xinyu Wang, and Elena L. Glassman. 2020. Interactive Program Synthesis by Augmented Examples. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 627–648. doi:10.1145/3379337.3415900

A Codebook

In this section, we give the full definition for the different entries in our codebook.

Code	**Description**
Preserving mental model	Participant wants to preserve their mental model of the current state of the protocol RE task .
Context research	Participant performs research into the context for the protocol RE task to inform the process, field choices, structure, etc.
Look for counters / magic numbers	Participant looks for well-known magic numbers / sequential counters.
Investigating and confirming hypotheses	Participant performs investigation and confirmation of hypotheses regarding the data.
User provided tests / data	Participant uses their own test and input data as part of the RE process.
Looking for patterns	Participant looks for patterns in the data.
Landmarking	Participant looks for landmarks in the data.
Preprocessing	Participant performs preprocessing of the data prior to examination.

Table 3: Codebook definitions for ProtocolREing processes

Code	Definition
Visualization	Participant wants changes to how data was presented or plotted.
Format / presentation of data	Participant wants changes to the format or presentation of the data.
Alternate plotting options	Participant wants alternate plotting options.
Final presentation of data for reporting	Participant wants feature related to output of results for reporting.
Alternative I/O	Participant wants means to move data into or out of the system.
External tool integration / interoperability	Participant wants integration or interoperability with an external tool.
User provided data	Participant wants to input a message or data and see how it is represented in the tool.
Generates Scaffolding	Participant wants tool to automatically generate code / interface for parsing data or IO with another tool.
Programmatic interaction with data structure	Participant wants to interact with the data in a programmatic way.
Focus	Participant wants to control the focus / layout / UI elements of the tool or data.
Searching	Participant wants to search data.
Bit Level View	Participant wants to view data at the bit-level.
Sorting	Participant wants to sort data.
Moving windows in and out of display	Participant wants to be able to use the interface across multiple windows, such as to facilitate use with other tools or to retain focus
Easy interface for drilling down	Participant wants tool to facilitate focusing on a subset of the data.
Hide parts of interface	Participant wants to show / hide parts of tool interface.
Filtering, general	Participant wants to select specific rows within the data to perform filtering actions.
Filtering by value	Participant wants to filter rows by the value of a field or fields.
Filtering by message sequencing	Participant wants to filter rows by a criteria and include messages with a relative offset (before or after) to those rows matching the criteria.
User filters data and reruns analysis	Participant wants to filter data and see how analysis changes.

Table 4: Tool Features codebook.

Code	Definition
Field Boundaries	Participant wants a feature related to or for manipulating field boundaries.
Automatic field boundary assignment	Participant wants automation to assist with assigning field boundaries.
Identifying connected features	Participant wants a feature identifying which pieces of information are related.
User defines connected features	Participant wants a feature allowing them to specify which pieces of information are related.
Automatic checksum management	Participant wants a feature related to automatic calculation or validation of checksum.
User choice in field boundaries	Participant wants to be able to set the field boundary themselves.
Bit boundaries	Participant wants boundaries at the bit level.
Filter / sorting applies across fields	Participant wants filtering and sorting capabilities that can be performed on one field should be able to apply to either reorder and filter data in other fields
Dotted vertical lines for visualizing breaks	Participant wants a visual indicator for visualizing splits before making the split.
Variable message size	Interface should support non-aligned field boundaries in messages which are variable length.
Variable field size	Participant wants field sizes to vary across messages.
Provide field boundary alternatives	Participants wants a feature which provides alternative field boundaries to one selected.
Automatic field boundary assignment	Participant wants a feature which automatically assigns field boundaries based on a condition.
Data type	Participant wants a feature related to or for manipulating data types.
Automatic data type assignment	Participant wants a feature related to automatic data type assignment.
User restricts data type and rerun analysis	Participant wants a feature related to automatic analysis with data type restrictions.
Additional supported encoding types	Participant wants a feature related to supporting additional text encoding formats.
User choice of endianness	Participant wants to be able to view / choose the endianness of the data types used.
User choice in mixed-type view in-line	Participant wants to select data and see it decoded as a bunch of different types simultaneously.
Provide data type alternatives	Participant wants tool to show alternative datatypes which are consistent with the data.
Decode as datatype	Participant wants to choose data and see it decoded as a particular datatype.
Decode as float	Participant wants to see data decoded as a float.
Decode as hex	Participant wanted to see data decoded as a hex.
Decode as ascii	Participant wanted to see data decoded as ASCII.
Additional supported data types	Participant wants additional data types beyond what was shown.

Table 4: Tool Features codebook.

Code	Definition
Meta data	Participant wants summary statistics, or metadata.
Timestamp	Participant wants to see timestamp when data was collected.
Summary statistics	Participant wants tool to calculate statistics such as min, max, median, mode for field values.
Identify data distribution	Tool should identify if data is drawn from a well known or user supplied distribution.
Histogram by field	Participant wants to see a histogram distribution for each field.
Show counts	Tool should display counts of messages / field values.
Confidence Metric	Automated system should provide a confidence measurement for inferred splits / types (OR) Interface should allow participant to record how confident they are in a piece of data.
Size of fields	Tool should give the size of fields based on field boundaries.
State-based modification	Participant wants features related to undo, redo, preview, or storing annotations.
Look back at previous states	Participant wants to be able to look at previous tool states for comparison.
Sticky notes / in-line comments	Participant wants to annotate data / interface with information, recording their mental model into the interface
Preview before change	Participant wants to see a preview of what will happen by making a change, comparing current state with next state
Undo button	Participant wants to undo last action, moving interface to previous state.
Impact of changes in field variance	Participant wants a tool feature allowing them to see the impact of changes on the variance of fields.
Displaying connected data	Participant wants features allowing brushing, or coordinated multi-view.
Coordinated Views / Brushing	Participant wants to select data and have related pieces of data highlighted throughout the interface.
Color coding	Participant wants data to be color coded by type or value such as all bytes in ASCII range shown with a fixed color.
Other	
Reason for automated choice	Participant wants information as to how the automated system made a choice.
Solution steps for automated decisions	Participant wants tool to provide specific steps that the automation took when developing a solution.
General customization	Particpant wants a feature related to general customization of the tool.

Table 4: Tool Features codebook.