Supplementary Material for BinXSight

1. Survey Methodology

1.1. Study Setup

We present the methodology of our survey in this section, including the design of the survey questionnaire, the recruitment, the survey procedure, the data analysis process, the detailed survey structure, and ethical considerations.

Design of the Survey Questionnaire. The survey questionnaire was developed in accordance with the exploratory motivation: *How useful and effective of BinXSight is in assisting reverse engineers with the understanding of unknown binaries*, to explore the practical value of it. To investigate this motivation, the survey seeks to evaluate the key dimensions of BinXSight's assistance capability. To this end, we carefully designed the following five aspects:

- *Meaningfulness and Helpfulness:* This dimension assesses whether the program summary generated provides valuable insights that aid reverse engineers in comprehending the functionality of the binary.
- *Minimal Superfluous Description:* This criterion examines whether the program summary contains unnecessary or irrelevant information that could detract from the focus of the binary's functionality.
- *Readability:* Readability evaluates how clear, concise, and easy-to-understand the program summary is for the participants.
- *Functionality Understanding:* This dimension focuses on whether the summaries effectively communicate the functionality of the binary.
- *Completeness:* Completeness assesses whether the summaries provide a thorough representation of the binary's functionality and core behaviors, including any critical information that may be necessary for effective analysis.

Based on the five aspects, we designed the following questions:

- *Q1 (Meaningfulness and Helpfulness):* Do you think the summary is meaningful and helpful for understanding the binary?
- *Q2 (Minimal Superfluous Description):* Do you think the summary has no or little superfluous descriptions that have nothing to do with the functionality of the binary itself?
- *Q3 (Readability):* Do you think the summary is clear, concise, and easy to understand the binary?
- *Q4 (Functionality Understanding):* Do you think the summary gives a good understanding of the functionality of the binary?
- *Q5 (Completeness):* Do you think the summary is complete (e.g., is there core functional information missing)?

The five aspects and corresponding questions outlined above form the foundation of our survey questionnaire, guiding the evaluation of BinXSight's effectiveness as a tool for reverse engineers. By addressing these key dimensions, we aim to gain comprehensive insights into the practical value that BinXSight offers in understanding unknown binaries.

Recruitment. We invited experts working in the field of reverse engineering and cybersecurity from both industry and academia to participate in the survey. We first started the recruitment by both personal contact and sending invitation Emails. For each contacted expert, we consulted whether they were willing to participate in our study, and provided a shopping card as a token of appreciation. In addition, we also used snowball sampling in our recruitment: at the end of each survey, we asked whether the participant could introduce other experts to join our study. This snowball sampling technique helped us expand our pool of participants in two ways: (1) by reaching experts within the same company or institution and (2) by including participants from different companies or institutions. Eventually, we have 27 study participants, including 15 seasoned reverse engineers from 3 leading security vendors, each with a minimum of 7 years of hands-on experience in reverse engineering, alongside 12 PhD candidates from 4 universities specializing in security research, all of whom possess at least 3 years of practical experience.

1.2. Procedure and Data Analysis

Survey Procedure. The survey was conducted through online meetings. During the process, the interviewers (i.e., authors of this paper) shared the screen to display the survey questionnaire and send the related materials to the participants. We began the survey process by collecting basic information about the participants, and then proceeded to score the program summaries based on the five questions in the questionnaire in a one-by-one manner (i.e., send the materials for the next binary after the current score is completed). We randomly choose 10 tuples of <*stripped binary (named in SHA256), first program summary, second program summary, VirusTotal URL, Project URL/Threat Reports*> from our test dataset results for each participant and query them for scoring, focusing on the five questions (i.e., Q1-Q5). *In which case, two summaries come from BinXSight, and {func-sum, func-code, sym-sum} with the best SIMILARITY score, and they are randomly assigned to either the first program summary or the second to eliminate bias.* We provide a tip: "*Please rate on a scale of 0 to 10,*

TABLE 1: 92 projects and 110 binaries in our dataset.

Project	# Binaries	Project	# Binaries
adns	3	aspell	1
awk	1	bash	1
bison	1	cflow	1
controlblock	1	cpio	1
cppi	1	cryptolocker	1
ctags	1	curl	1
datamash	1	db-bench	1
direvent	1	ecal-rec	1
enscript	1	st-device-sdk-c	1
ffmpeg	1	freetype	1
gcal	1	gdbm	3
gh0st	3	grep	1
gzip	1	kvrocks	1
less	1	a2ps	1
audioflux	1	glpk	1
gmp	1	gnudos	1
gsasl	1	gsl	1
guile	1	idn	1
libjpeg	1	lightning	1
magickwand	1	libmicrohttpd	1
mpfr	1	nettle	1
osip	2	libpng	1
readline	1	sqlite	1
libtasn1	1	libunistring	1
libxml2	1	linux-ddos	1
linux-encoder	2	linux-wirenet	1
llama2.c	2	macchanger	1
make	1	masscan	1
minhook	1	mirai	2
nano	1	nano-node	1
objcopy	1	objdump	1
openssl	1	patch	1
poke	1	ransom-wannaren-decryption	1
ransomware-wannacry	1	ransomware-xdata	1
rbot	1	readelf	1
replay-sorcery	1	sed	1
ssdeep	1	openssh	3
sshd-malware	5	tar	1
tcpdump	1	tic80	1
time	1	tmux	1
tortoisegitmerge	1	tortoisegitproc	1
ujson	1	units	1
unzip	1	wdiff	1
wget2	1	whisper	3
winsparkle	1	xmrig	1
xorriso	1	z3	1

with higher scores indicating better quality." to guide the scoring of each question for the two summaries. Due to the total of 10 binaries for each participant for analyzing (see Appendix 1.3) and scoring, the survey process is allowed to be interrupted and resumed. After the survey study, we collected 270 $\times$ 2 scores for each question. The survey was started in January 2025 and finished in March 2025.

Data Analysis. Since the survey is conducted in the form of ratings, we can directly count the scores for each question. Additionally, for further interpretation of the scores after the statistics, we conducted the interview through email.

1.3. Survey Structure

The survey starts with an introduction, where objectives are outlined. We then collect basic information about the participants' work experience and role within their organizations. In the next stage, the survey delves into the specific scoring process. Specifically, the scoring procedure for each binary requires participants to follow the steps listed below:

1) ***Independent Analysis:*** Require participants to perform independent reverse engineering within an hour. In this step, we only provide the VirusTotal URL to participants as the initial searchable material for assisting reverse engineering. We also allow participants to search for available resources from the internet. This step ensures that their initial understanding of the binary is based solely on their skills.
2) ***Reference:*** We further provide the project URL or threat reports of the binary to supplement their understanding after the initial independent analysis.
3) ***Scoring:*** Eventually, two corresponding program summaries are provided for scoring for the five questions.

After all 10 binaries have been scored, we conclude the survey process for participants.

1.4. Ethical Considerations

Before each survey, we explained our data collection goals, the anonymization process for the collected data, and began the process only after obtaining permission. All participants had approval from their respective organizations to participate in the survey. They were fully aware of our data collection process and purpose, and consented to our data collection. Moreover, all collected data were securely stored on encrypted cloud storage, in accordance with the General Data Protection Regulation (GDPR) and previous studies [1].

2. Projects and Binaries in Dataset

Table 1 presents the 92 projects that we used in our dataset, which contains 110 unique stripped binaries in total. These projects cover diverse fields, such as network communication, remote control malware, ransomware, DDoS virus, gaming, machine learning, and so forth, across various platforms, including Windows, Linux, and macOS.

3. Comparative Examples

The comparative examples, shown in Table 2, demonstrate that BinXSight better helps reverse engineering understand binaries without the need for manual reverse engineering. It comprehensively summarizes the overall functionalities of binaries, whether for benign samples or malicious samples. In contrast, the baseline methods struggle to capture the functionalities needed for users to quickly understand unknown binaries.

References

[1] E. Kenneally and D. Dittrich, "The menlo report: Ethical principles guiding information and communication technology research," *Available at SSRN 2445102*, 2012.

TABLE 2: The summaries for comparative examples by BinXSight and the baselines (with best SIMILARITY scores).

Binary	BinXSight	Baseline	Reference
ctags (benign software)	The binary program is a versatile tool for generating and managing code tags across multiple programming languages. It parses source code to extract information such as functions, variables, and classes, creating tag files that facilitate code navigation and searching. Additionally, it supports structured data processing and validation, particularly for formats like XML and JSON, and includes features for schema validation and XPath evaluation. ***The program allows for extensive customization in tag generation and supports various output formats, making it a comprehensive solution for code indexing and structured data management.***	**The binary program appears to be a complex parser and validator for XML and RelaxNG schemas.** It includes various functions that handle different aspects of XML processing, such as parsing elements, attributes, and namespaces, as well as managing error reporting and validation against defined schemas. The program supports a range of XML constructs, including elements, attributes, and complex types, and it implements mechanisms for handling namespaces and validation rules. Functions are defined to manage the parsing of specific constructs, such as classes, traits, and methods, while also providing error handling for various validation scenarios. The program also includes support for XInclude, allowing for the inclusion of external XML documents. Overall, it serves as a robust tool for XML schema validation and manipulation, ensuring compliance with defined standards and providing detailed error reporting for any discrepancies encountered during processing.	https://github.com/universal-ctags/ctags
nano-node (benign software)	The binary program functions as a decentralized cryptocurrency node and management system, facilitating transaction processing, account management, and blockchain integrity maintenance. It supports various network protocols and enables user interaction through command line and API via RPC. Key features include transaction verification, voting mechanisms, block confirmations, and synchronization among nodes, ensuring network security and efficiency. The program also incorporates modules for wallet operations, HTTP header management, and JSON-RPC request handling, while utilizing database backends like LMDB and RocksDB for optimized data storage and access. ***Overall, it serves as a robust backend for cryptocurrency applications, enabling real-time communication and efficient management of decentralized digital currency operations.***	**The binary program is a complex multi-threaded application that utilizes various system calls and library functions to manage threads, perform file operations, and handle network communications.** It includes functions for creating and managing threads using pthreads, manipulating file descriptors, and performing input/output operations on files. The program also implements error handling and memory management, including dynamic memory allocation and deallocation. Additionally, it features locale management, string manipulation, and mathematical computations. The binary is designed to handle concurrent operations efficiently, utilizing mutexes and condition variables for synchronization. It also includes functions for signal handling and process management, indicating its capability to interact with the operating system at a low level. Overall, the binary serves as a robust framework for executing tasks that require multi-threading and system-level operations.	https://github.com/nanocurrency/nano-node
z3 (benign software)	***The binary program is a sophisticated SMT solver designed to efficiently address complex logical and mathematical problems.*** It supports a wide range of logics, including arithmetic, Boolean logic, and bit vectors, and is capable of handling quantifiers and constraints. The program features multiple solving strategies, incremental solving, and parallel processing to enhance performance on large-scale problems. It also offers model generation, proof generation, and unsatisfiable core identification, making it ideal for applications in formal verification, program analysis, and automated theorem proving. Additionally, it includes functionalities for managing logical expressions and optimizing reasoning tasks across various computational contexts.	**The binary program is a complex software component that primarily handles file input/output operations, memory management, and exception handling.** It utilizes C++ standard library features, including streams for reading and writing data, and employs various mathematical functions for computations. The program includes functions for managing threads and mutexes, ensuring thread safety during concurrent operations. It also implements error handling mechanisms, throwing exceptions for various error conditions, such as memory allocation failures and invalid arguments. The binary is designed to support operations on data structures, including lists and trees, and provides functionality for string manipulation and formatting. Additionally, it integrates system-level calls for process management and signal handling. Overall, the binary serves as a robust foundation for applications requiring efficient data processing, error management, and multi-threading capabilities.	https://github.com/Z3Prover/z3
linux-wirenet (malware, collected in the wild, 46/64 AV Engines)	***The binary program functions as a comprehensive surveillance and control tool, primarily designed for malicious purposes.*** It includes keylogging capabilities to capture user input, manages and extracts credentials from various applications like Mozilla and Pidgin, and facilitates remote system control through a backdoor mechanism. The program can perform file operations, including downloading and managing files, and operates as a proxy server for secure network communication. Additionally, it captures screen content and processes images, while employing AES encryption for secure data transmission. ***Overall, the program integrates multiple functionalities for monitoring, data exfiltration, and remote manipulation of compromised systems.***	**The binary program is a comprehensive utility designed for file management, network communication, and system interaction, primarily in a Unix-like environment.** It includes functions for file operations such as reading, writing, and deleting files, as well as managing directories and file permissions. The program also facilitates network connections through various protocols, including SOCKS and HTTP, enabling data transfer and communication with remote servers. Additionally, it incorporates encryption and decryption functionalities, particularly using AES and RC4 algorithms, to secure data transmission and storage. The binary features user authentication mechanisms, including retrieving and decrypting stored passwords from popular browsers like Chrome and Firefox. It also supports system information retrieval, such as process IDs and memory status, while providing capabilities for mouse and keyboard event handling in an X11 environment. Overall, the binary serves as a multi-functional tool for managing files, network operations, and system resources efficiently.	https://www.virustotal.com/gui/file/35ff79dd456fe3054a60fe0a16f38bf5fc3928e1e8439ca4d945573f8c48c0b8
xmrig (malware, collected in the wild, 43/65 AV Engines)	***The binary program is a cryptocurrency mining application specifically designed for mining Monero using CPU resources.*** It supports multiple mining algorithms, particularly the Cryptonight series, and allows users to configure various parameters such as pool addresses, user credentials, and CPU usage. The program facilitates efficient communication with mining pools, ensuring secure data transmission through TLS encryption. It features a multi-threaded architecture for optimal resource allocation, manages asynchronous I/O operations, and includes logging and configuration management for monitoring mining performance. Overall, the program provides a comprehensive and user-friendly solution for cryptocurrency mining, focusing on performance and security.	**The binary program primarily functions as a complex data processing and encryption tool, heavily utilizing the AES encryption algorithm.** It features numerous functions that handle various tasks, including memory management, data validation, and condition checks. Many functions serve as wrappers or forwarders to a core function, 'FUN_00465d80', which is pivotal to the program's operations. The program also manages linked lists and dynamic memory, ensuring data integrity and security through rigorous checks and balances. Additionally, it processes input parameters for network communication, handles errors, and performs logging for debugging purposes. The binary is designed to efficiently manage resources, execute encryption tasks, and maintain the state of various data structures, making it suitable for applications requiring secure data handling and processing. Overall, the program's architecture reflects a focus on modularity, error handling, and efficient data manipulation.	https://github.com/xmrig/xmrig