International Journal For Technological Research In Engineering
Volume 2, Issue 8, April-2015 *ISSN (Online): 2347 - 4718*

A REVIEW ON PROTOCOL REVERSE ENGINEERING

Nikita Bakshi[1], Yamini Pathania[2], Poonam Kumari[3]
Department of CSE, Chandigarh, India

Abstract: Protocol reverse engineering has been a manual process that is considered tedious, time-consuming and error-prone. To tackle this limitation, many solutions have been proposed recently to allow automatic protocol reverse engineering. Application-level protocol specifications are helpful for many safety measures, including detection and intrusion prevention that performs traffic normalization, deep packet inspection and penetration testing that produces network inputs application to expose potential vulnerabilities. Though, current practice in deriving protocol specifications is mostly manual. In this paper, we present a tool for reverse engineering the protocol message formats of an application to trace network automatically. A fundamental property of the Discoverer is that it works independently of the idioms inferring protocol usually seen in multiple message formats application-level protocols.
Keywords: Software, Reverse Engineering, Protocol, Polyglot.

I. INTRODUCTION

A. Reverse Engineering

The process of reconstructing a design by checking a final product. Reverse Engineering is same in both software and hardware. Reverse engineering is the process of determine the technological principles of a object, device, or system through analysis of its function, structure, and operation. It generally involves disassembling something (a, electronic component, computer program, or biological, chemical, or organic matter, mechanical device) and analyzing its components and running in detail, just to re-create it. Reverse Engineering is used for maintenance or to create a new device or program that does the same thing, without understanding the original or using, or simply to duplicate it. Reverse engineering starts from the analysis of hardware for commercial or military advantage. The objective is to reduce product design decisions end with little or no additional knowledge about the procedures involved in the original production. Reverse engineering has many applications it can be used as a learning tool as a way to make new, easy to use products that are cheaper than what's currently on the market and for making software interoperate more productively or to bridge data between different operating systems or databases; and to uncover the undocumented features of commercial products.

B. Protocol Reverse Engineering

In information technology, a protocol is the unique set of rules that end points in a telecommunication connection utilizes when they communicate. Accordingly, the difficulty in protocol reverse-engineering is it can be partitioned into two sub problems, state-machine and message format reverse-engineering. The message formats are generally reverse engineering through a manual process of tuff, which is referred as message processing protocol implementations, but now research proposed many solutions automatic. Normally, these automatic approaches either emulate the protocol implementation tracing the message processing or group observed messages into clusters using various clustering analyses. Less work has been done on reverse-engineering of state-machines of protocols.

II. APPLICATION OF PROTOCOL REVERSE ENGINEERING

A. Polyglot - With Polyglot, we provide the first binary analysis technique for automatic protocol reverse engineering. Earlier work on automatic protocol reverse engineering find out protocol information from network traces. Polyglot grasps the accessibility of a program by implementing the binary protocol. Polyglot uses a different intuition every time, the way that an implementation of the protocol proceeds the received application data discloses a wealth of information about the style of a accepted message. Compared to network traces, which only hold syntactic information, program binaries also hold semantic data about how the program processes and implements on the protocol data. In addition, they are a very important source of information about the performance of the protocol.

B. Dispatcher - Dispatcher is our newest automatic protocol reverse engineering tool and superseeds Polyglot. Polyglot techniques are included, also adds three important contributions. Firstly , it executed buffer deconstruction , a special technique to get out the format of the messages that are sent from the application of the implementation of the protocol , when Polyglot only draw format of the messages received from the application. Secondly, the dispatcher also extracts semantic information on the fields of messages received and sent. For example, it can identify field is a timestamp, a filename, or an IP address. Finally Dispatcher able to reverse engineer encrypted protocols by identifying the buffers that hold message received unencrypted after it has been decrypted by the program and buffers that keep unencrypted message to be sent before the request is encrypted. Then apply the automatic protocol reverse engineering techniques on these buffers.

C. Replayer - The ability to accurately replay application protocol dialogs is useful in many security-oriented applications, such as replaying an exploit for forensic analysis or demonstrating an exploit to a third party. A central challenge in application dialog replay is that the dialog intended for the original host will likely not be

accepted by another without modification. For example, the dialog may include or rely on state specific to the original host such as its hostname, a known cookie, etc. In such cases, a straight-forward byte-by-byte replay to a different host with a different state (e.g., different hostname) than the original observed dialog participant will likely fail. These state-dependent protocol fields must be updated to reflect the different state of the different host for replay to succeed.

III. LIMITATIONS

There are two main fundamental limitations:

A. Trace Dependency - The format generated by any tool that operates only on the trace is limited by the diversity of traffic seen in the trace. If certain message formats never occur in the trace, or if certain variable fields never take more than one value in the trace, it is impossible for such a tool to infer those message formats or identify those fields as variable fields.

B. Pre-Defined Semantics: Only a set of pre-defined semantics can be inferred. Since it is not possible to find all the possible semantics of all fields just from a trace, the best one can hope for is to have an extensible framework where new semantic modules can be added as desired.

IV. TOOL USED

A. Disassembler - A disassembler attempts to dissect a binary executable into human readable assembly language. The disassemble software reads the raw byte stream output from the processor and parses it into groups of instructions most commonly used disassemble is IDA Pro IDA. It is a multiprocessor, multioperating system, interactive disassembler. Most powerful feature is FLIRT-Fast Library Identification and Recognition Technology.

B. Debugger - The most commonly used debugger is ollydbg. It is often used by crackers to crack software made by other developers. For cracking and reverse engineering, it is often the primary tool because of its ease of use and availability; any 32 bit executable can be used by the debugger can be edited in bit code/assembly in real time. It is also useful for programmers to ensure that their program is running as intended. Furthermore it can be used for malware analysis purposes as well.

C. WireShark - Wireshark (formerly known as Ethereal) is a free, open source network protocol analyzer. It can be useful for capturing data packets on the network wire for later analysis to understand how proprietary multimedia networking protocols operate.

D. Internet Junk buster - The Internet Junkbuster is a useful tool for filtering web requests that a web browser makes to sites that are known to do little more than serve advertisements. However, it is also a highly configurable, general HTTP proxy server and can be leveraged as sort of a poor man's network protocol analyzer. The IJB can be used in place of a full-fledged network protocol analyzer when the task at hand is to understand how data, encapsulated entirely in HTTP requests and responses, is formatted. The master Junkbuster configuration file has a configuration setting named debug. This setting allows the user to log data such as connection status, HTTP responses and requests, and even the full data transferred.

V. FUTURE WORK

Protocol reverse engineering is a highly manual process today, which is still suffered through because of the immense value of protocol knowledge. We have developed Discoverer, a tool that aims to automate this reverse engineering process. Discoverer leverages recursive clustering and type-based sequence alignment to infer message formats. We have demonstrated Discoverer can infer message formats effectively for three network protocols, CIFS/SMB, RPC, and HTTP. In the future, we intend to enhance our semantic inference, research on the inference protocol state machine, explore the sense of using the analysis of reverse engineering software to the specifications of both networks and input file and apply the protocol specifications reverse engineering for real-world applications. Many of the limitations above are due to the limited information available from network traces. To tackle these limitations and achieve a better reverse-engineered protocol specification, we can use program analysis to gain more information and insight into the parsing and processing of the input in the program. For instance, we may easily identify two consecutive bytes as a WORD (i.e., a two-byte integer) from run-time analysis by observing that they are processed as a WORD throughout the execution. We have focused on reverse engineering network protocols in Discoverer; it would be useful to reverse engineer the input specifications for file-based applications, since we have seen a significant growth in file-based attacks.

REFERNCES

[1] Li, Xiangdong, and Li Chen. "A survey on methods of automatic protocol reverse enginesering." 2011 Seventh International Conference on Computational Intelligence and Security (CIS) IEEE, 2011.
[2] Pan, Fan, et al. "Efficient Protocol Reverse Method Based on Network Trace Analysis." JDCTA: International Journal of Digital Content Technology and its Applications 6.20, 2012.
[3] Cui, Weidong, Jayanthkumar Kannan, and Helen J. Wang. "Discoverer: Automatic Protocol Reverse Engineering from Network Traces." USENIX Security. 2007.
[4] D. Cooper, B. Khoo, B. R. von Konsky, and M. Robey.Java implementation verification using reverse engineering. In ACM International Conference Proceeding Series, ACM, 2004.
[5] Shevertalov, Maxim, and Spiros Mancoridis. "A reverse engineering tool for extracting protocols of networked applications, 14th Working Conference on Reverse Engineering. IEEE, 2007.
[6] Wang, Zhi, et al. "ReFormat: Automatic reverse

engineering of encrypted messages." Computer Security–ESORICS 2009. Springer Berlin Heidelberg, 2009.
[7] Cipresso, Teodoro, Software Reverse Engineering Education.SJSU Master's Thesis, 2009.
[8] Eldad Eilam. Reversing: Secrets of Reverse Engineering.Indianapolis: Wiley, 2005.
[9] E. J. Chikofsky and J. H. CrossII, ReverseEngineering and Design Recovery: A Taxonomy, IEEE Software,Jan 1990.
[10] Juan Caballero, Pongsin Poosankam, Christian Kreibich, and Dawn Song. In Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), November 2009.
[11] James Newsome, David Brumley, Jason Franklin, and Dawn Song. In Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS), October 2006.

