An Automatic Protocol Reverse Engineering Approach from the Viewpoint of the TCP/IP Reference Model

Young-Hoon Goo
Advanced KREONET Center
Korea Institute of Science and Technology Information
Daejeon, Korea
gyh0808@kisti.re.kr

Kyu-Seok Shim
Advanced KREONET Center
Korea Institute of Science and Technology Information
Daejeon, Korea
kusuk007@kisti.re.kr

Ui-Jun Baek
Computer Information and Science
Korea University
Sejong, Korea
pb1069@korea.ac.kr

Jee-Tae Park
Computer Information and Science
Korea University
Sejong, Korea
pjj5846@korea.ac.kr

Mu-Gon Shin
Computer Information and Science
Korea University
Sejong, Korea
tm0309@korea.ac.kr

Myung-Sup Kim
Computer Information and Science
Korea University
Sejong, Korea
tmskim@korea.ac.kr

***Abstract*—Protocol reverse engineering represents a very powerful and important tool for network management and security. To cope with the emergence and evolution of rapidly increasing numbers of unknown protocols, automation is of great importance. Many methods for supporting the automation of the various steps for protocol reverse engineering have been investigated; however, there has been no method to automate the analysis of the target network environment. Most methods are designed only for application layer protocols, and all others are designed for specific environments. Given any unknown communication, we must be able to infer the structure of the protocol. However, there has been no research on automatic reverse engineering of protocols when both the protocol and the target network environment are entirely unknown. Here, we propose an automatic protocol reverse engineering approach that is designed to be generally applicable, regardless of the specific network environment. We demonstrate the feasibility of the proposed approach by applying it to several protocols in various layers of the TCP/IP reference model.**

Keywords—protocol reverse engineering, TCP/IP reference model, genericity

I. Introduction

Protocol reverse engineering, which is the process of inferring a specification for an unknown protocol, is a potent and powerful tool for network management and security. For instance, by securing the specifications of unknown protocols, network administrators can effectively establish network policies such as network planning, Quality of Service (QoS) provisioning, and control of network usage for specific protocols, by classifying the traffic generated from these unknown protocols. It can also be helpful in numerous domains of network security, including malware protocol analysis, simulation of network protocols, vulnerability analysis, smart fuzzing, and the provision of useful information to intrusion detection and prevention systems (IDSes/IPSes) [1]. In particular, protocol reverse engineering has become increasingly important over the last decade, in light of the proliferation of the Internet-of-Things (IoT) devices, intensifying cyber-conflicts across countries, and constant security incidents. In these situations, many protocols are unknown, and they are rapidly evolving and increasing in number.

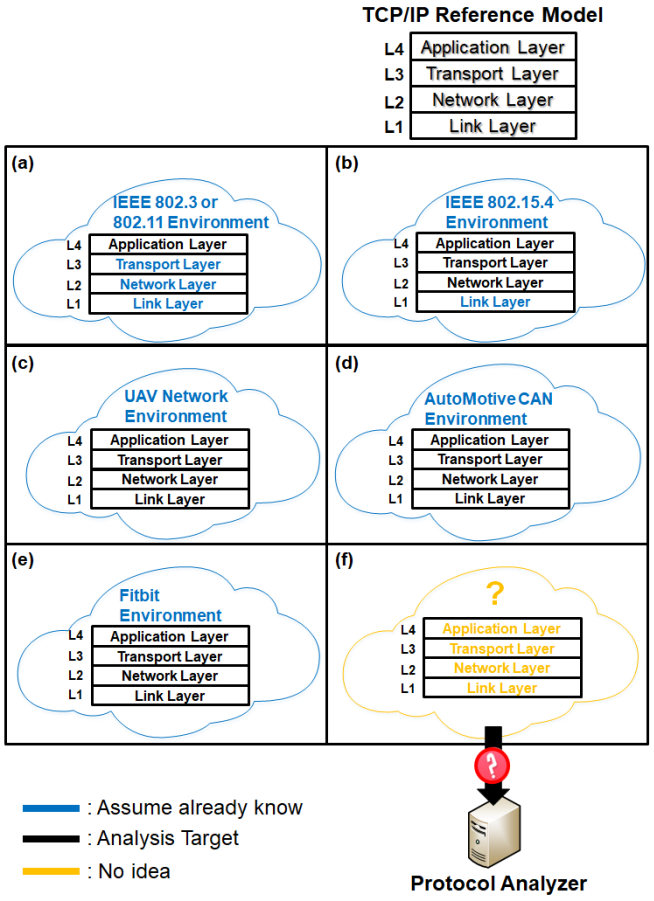

Fig. 1. Specific environments that previous studies have focused on (a)–(e) and entirely unknown environment that we discuss (f)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean Government (MSIT) (No.2018-0-00539, Development of Blockchain Transaction Monitoring and Analysis Technology), in part by the Technology Innovation Program (No. 20008902, Development of SaaS SW Management Platform based on 5 Channel Discovery technology for IT Cost Saving) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea), and in part by Korea Institute of Science and Technology Information (KISTI).

Traditionally, protocol reverse engineering is a time-consuming, tedious and error-prone task. To cope with the emergence and evolution of rapidly increasing unknown protocols in today's high-speed and complicated network environments, the automation of protocol reverse engineering is essential. Thus, many automatic protocol reverse engineering methods have been introduced. However, all existing methods are designed for specific environments; no matter how excellent a method is, it cannot be applied to other network environments. There has been no research on automatic protocol reverse engineering when both the protocol and the target network environment are entirely unknown.

Here, we propose a protocol reverse engineering approach that is automated and designed to be generally applicable, regardless of the specific network environment. This follows a step-by-step approach from the lower-level to upper-level layers of the TCP/IP reference model and ascertains whether there is any known information. Using this general approach, automatic protocol reverse engineering can be performed without any knowledge of the target network environment. We demonstrate the feasibility of the proposed approach by applying it to several protocols in various layers of the TCP/IP reference model.

Section 2 describes the rationale behind the proposed approach, and Section 3 explains the proposed approach. Then, Section 4 presents the experimental results. Finally, Section 5 concludes this paper.

II. Rational of Proposed Approach

A communication between two devices needs to follow some protocol, and actually consists of several protocols called a protocol family, based on the TCP/IP reference model, shown in (f) of Fig. 1. The TCP/IP reference model is a conceptual model, which is composed of four distinct layers: the link layer (layer 1), network layer (layer 2), transport layer (layer 3), and application layer (layer 4) [2], [3].

Meanwhile, existing protocol reverse engineering methods have made considerable progress in automating the various steps required for protocol reverse engineering: from collecting data up to representing an approximate inferred specification. However, all of these methods assume that they already know the target network environment and analyze a protocol in a particular layer in the TCP/IP reference model. Namely, none of them are automated in terms of the network environment analysis. Most previous studies [4]–[6] have only focused on inferring the application layer protocol over the IEEE 802.3 Ethernet network (Fig. 1, (a)). Some previous studies have considered certain environments other than IEEE 802.3 Ethernet. For example, Wang *et al*. [7] and Li *et al*. [8] targeted the application layer protocols in IEEE 802.11 wireless environments (Fig. 1, (a)), and WasP [9] targeted customized protocols built on top of the MAC layer of IEEE 802.15.4 (Fig. 1, (b)). Ji *et al*. [10], READ [11], and Fereidooni *et al*. [12] focused on reverse engineering the flight control protocols of unmanned aerial vehicles (UAVs), the automotive protocols of controller area network (CAN), and the Fitbit protocol used for wearable fitness trackers, respectively (Fig. 1, (c)–(e)). These methods cannot be applied to other network environments. In more detail, Pext [13] and AutoReEngine [14] use flow information to infer a protocol finite state machine. FieldHunter [15] uses the TCP push flag to assemble packets to message unit. When inferring a protocol that does not use TCP or UDP as a carrier protocol, the above methods are useless. If a target unknown protocol is a link layer protocol, a network layer protocol, or a transport layer protocol, these methods cannot define the flow, so behavior inference, such as state machine and message order between two hosts, is impossible.

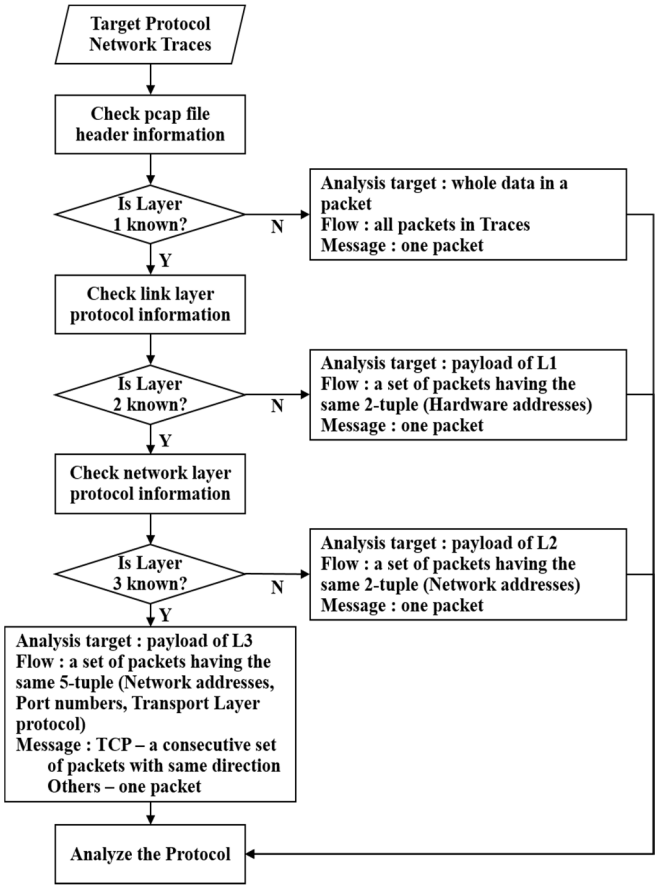

Fig. 2. Flowchart of proposed method

Consequently, given any communication, we must be able to perform protocol reverse engineering without considering the particular network environment and designation for a particular layer. To make this possible, and to maximize the utilization of algorithms of existing methods, network environment analysis (such as the configuration of the message unit, flow unit, and range of data in a packet) should be automated.

III. Proposed Approach

A. Key Insight

In general protocol layering, a protocol family has a hierarchical structure; each upper-level protocol is supported by one or more lower-level protocols. In other words, each layer at the sending site uses the services of the layer immediately below it. The sender at the higher-level layer uses the services of the middle-level layer. The middle-level layer uses the services of the lower-level layer. The lower-level layer uses the services of the carrier [17].

The TCP/IP reference model is also layered and became the dominant commercial architecture because it was used and tested extensively in the Internet. The key insight of the proposed method is that, if information about the protocols of

the lower-level layers is available, it is easy to analyze protocols of the upper-level layers by using this information. Therefore, the proposed method takes a stepwise approach, from layer 1 to layer 4 of the TCP/IP reference model. Then, it decides how to set the message unit, flow unit, and range of data to analysis in each packet.

It is momentous to automatically determines which protocol layer to analyze, i.e., redefining message, flow, range of data to analyze in a packet; The definition of message, i.e., protocol data unit (PDU), depends on the protocol layer, and the definition of flow must be re-determined to infer the behavior of the protocol. This allows us to apply protocol reverse engineering universally.

B. Methodology

Fig. 2 presents the flowchart of the proposed approach. It first checks whether the protocol of the link layer is already known, by inspecting the pcap file header information. If the link layer's protocol is unknown, then none of the physical or logical addresses information is known. Therefore, the flow unit is set to all packets of the network traces, the message unit is set to one packet, and protocol reverse engineering is performed on the entire data of each packet.

Otherwise, the method moves on to the next step. It checks whether the protocol of the network layer is already known, by inspecting the link-layer information. In this step, if the network layer's protocol is unknown, then the flow unit is set to the set of packets having the same two-tuple: the source and destination physical address that can be obtained from the link layer information. The message unit is set to one packet. Then, protocol reverse engineering is performed on the payload of the link layer's protocol for each packet.

Otherwise, the method moves on to the next step, and checks whether the protocol of the transport layer is already known, by inspecting the network layer information. If the transport layer's protocol is unknown, then the flow unit is set to the set of packets having the same two-tuple: the source and destination logical address that can be obtained from the network layer information. The message unit is set to one packet. Protocol reverse engineering is then performed on the payload of the network layer's protocol of each packet.

Otherwise, the link layer, network layer, and transport layer information are all known, and so the flow unit is set to the set of packets having the same five-tuple (comprising the source and destination logical address, source and destination port, and the transport layer protocol) as the general definition of flow. In this step, if the transport layer protocol is TCP, then the message unit is set to the series of consecutive packets in the same direction. Otherwise (the transport layer protocol is UDP), the message unit is set to one packet. Protocol reverse engineering is performed on the payload of the transport layer's protocol for each packet.

C. Application of Proposed Approach

The proposed approach can achieve a higher utilization if the more types of modules that identify known protocols are implemented for each of the layers.

Fig. 3 illustrates how to apply the proposed approach. This is an example of implementing only the three modules that can identify IEEE 802.3 Ethernet, IP, and TCP, as modules that identify known protocols corresponding to the link layer, network layer, and transport layer, respectively.

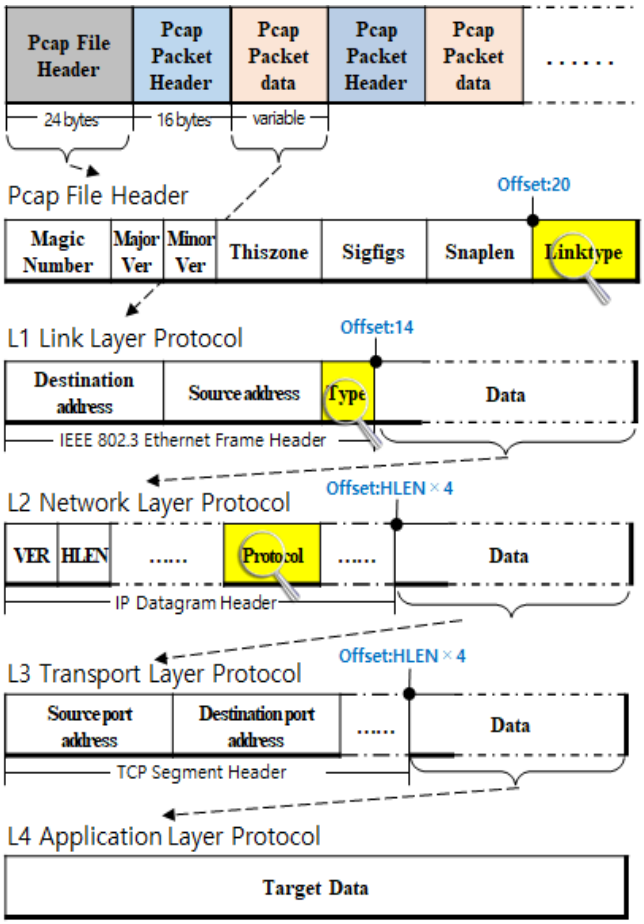

Fig. 3. Application of proposed method

The pcap file header contains a member variable named Linktype, which contains information on what the link layer's protocol is, and so the proposed approach starts by checking this member variable. In this case, it checks whether the link layer is IEEE 802.3 Ethernet. If the link layer is not IEEE 802.3 Ethernet, then the analysis target for protocol reverse engineering is the entire data in each packet, the message unit is set to one packet, and the flow unit is set to all packets of the network traces. Otherwise, the next steps are followed in sequence as described above. The method checks whether the network layer is IP by inspecting the *Type* field of IEEE 802.3 Ethernet, and subsequently checks whether the transport layer is TCP by inspecting the *Protocol* field of IP. Thus, inferring an unknown protocol becomes easier, and automatic protocol reverse engineering can be applied universally, regardless of the specific environment.

IV. Experimentation

A. Tool and Protocols

To demonstrate the feasibility of the proposed approach, we conducted experiments by applying it to a protocol reverse engineering tool that we previously developed [18]. This tool can extract not only field formats and message formats, but also flow formats, which are the main types of flow of the target protocol, for an unknown protocol. It does this by using the contiguous sequential pattern algorithm, which is an algorithm for finding frequent patterns, three times. The detailed inference method of this tool is beyond the scope of this paper, and so it is not described here.

The experiments were conducted on five protocols: HTTP and DNS for the application layer protocols, ICMP for the transport layer protocol, ARP for the network layer protocol, and MIL-STD-188-220c for the link-layer protocol. ICMP is a network layer protocol, but it was treated as a transport layer protocol, because it generally operates above the IP layer. Table 1 presents the quantitative information of the dataset for the five protocols in flow, packet, and byte.

TABLE I. QUANTITATIVE INFORMATION OF THE DATASET FOR FIVE SELECTED PROTOCOLS

Protocol	Flows	Packets	Bytes
HTTP	359	3,841	4,670
DNS	2,170	4,349	740
ICMP	127	1,001	70
ARP	226	15,000	960
MIL-STD-188-220c	1	900	280

B. *Experimental Results*

Table 2 summarizes the experimental results, in terms of coverage, defined as the ratio of messages that could be analyzed by the extracted message formats to the total messages. By applying the proposed approach, we could automatically extract the message formats for all five protocols with 100% coverage. In contrast, all other existing methods can only be applied to protocols for a particular layer of the TCP/IP reference model. AutoReEngine [14] could only infer the application layer protocols, as shown in Table 2. Netzob [19] could infer all five protocols, but only when provided with a manual designation of a specific layer. Moreover, application layer protocols that are not over TCP or UDP, and transport layer protocols that are not over IP, cannot be inferred by Netzob [19].

TABLE II. EXPERIMENTAL RESULTS IN TERMS OF COVERAGE

Protocol	[18] with the proposed method	AutoReEngine [14]	Netzob [19]
HTTP	100% (automatic)	100%	31.8% (manual)
DNS	100% (automatic)	53.7%	87.9% (manual)
ICMP	100% (automatic)	- (cannot)	100% (manual)
ARP	100% (automatic)	- (cannot)	98.7% (manual)
MIL-STD-188-220c	100% (automatic)	- (cannot)	100% (manual)

The reason AutoReEngine [14] does not have 100% coverage for some protocols is that it does not have the ability to infer dynamic fields. Therefore, there are some message

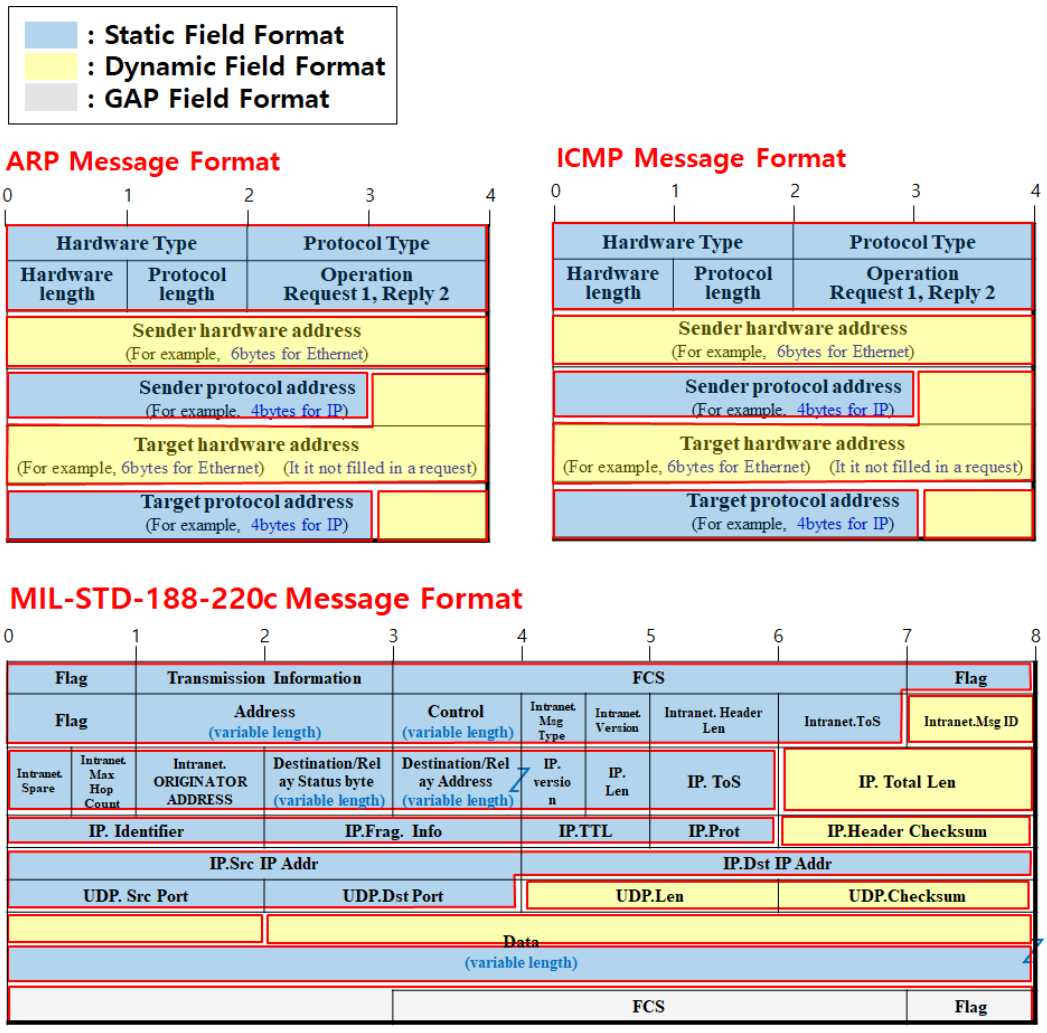

Fig. 4. Representative extracted message formats for layer 1, 2, and 3

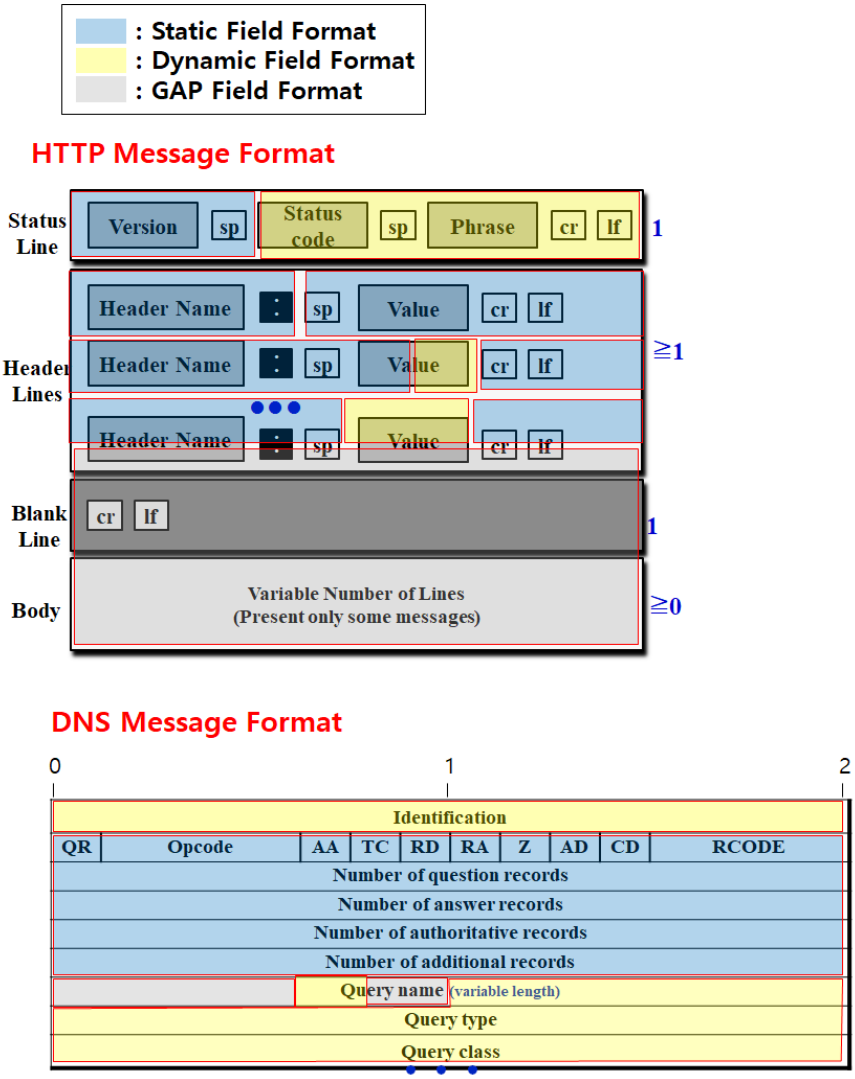

Fig. 5. Representative extracted message formats for layer 4 (HTTP, DNS)

formats that could not be extracted. Netzob [19] extracts a very large number of message formats that can cover only one message, meaningless message formats; eliminating the meaningless message formats reduces the coverage.

Fig. 4 and Fig. 5 show the representative extracted message formats of HTTP, DNS, ICMP, ARP, and MIL-STD-188-220c. For each protocol, the figure compares extracted message formats with the actual specifications. It is apparent that almost all parts of the actual specification can be extracted as static and dynamic fields, but not the gap fields.

As a result, we have shown that it is possible to infer protocols of any layers of the TCP/IP reference model by applying the proposed approach.

C. *Discussion*

Protocol reverse engineering methods can be divided into network trace-based analysis and application-based analysis. Protocol encryption, such as using the VPN or SSH protocol, thwart protocol reverse engineering. In fact, network trace-based protocol reverse engineering method cannot infer a specification of an encrypted protocol. To achieve this, we should use application-based analysis, using execution traces or source code; Reformat [20] and Dispatcher [21], that uses application-based analysis, are the only methods that can infer a specification of an encrypted protocol. However, because a program's binary or source code is not available in many cases, network trace-based methods have been the subject of the most active research in recent decades.

V. Conclusion and Future Work

In this paper, we have proposed an automatic protocol reverse engineering approach from the viewpoint of the TCP/IP reference model. The novelty of this approach is that it is generally applicable, regardless of the specific network environment. This is important, because it automates the analysis of the network environment, unlike existing methods.

Using our automatic network environment identification, the utilization of existing protocol reverse engineering tools can be improved. In addition, it can be extended by adding modules that identify various known protocols.

Through experiments on several protocols, in all layers of the TCP/IP reference model, we have demonstrated the feasibility of the proposed approach in various network environments.

Encryption and existence of optional field sequences are factors that make inferring a specification difficult. We consider developing a hybrid method that uses both execution

traces and network traces to combat these problems. Further, we also plan to conduct further research on real-time reverse engineering, big-data processing, and maintenance of extracted specifications.

REFERENCES

[1] J. Duchene, C. L. Guernic, E. Alata, V. Nicomette, and M. Kaaniche, "State of the art of network protocol reverse engineering tools," J. Computer Virology and Hacking Techniques, vol.14, no.1, pp. 53-68, Feb. 2018.

[2] R. Braden, RFC 1122, IETF, http://tools.ietf.org/html/rfc1122, Oct. 1989.

[3] R. Braden, RFC 1123, IETF, http://tools.ietf.org/html/rfc1123, Oct. 1989.

[4] S. Kleber, H. Kopp, and F. Kargl, "NEMESYS: network message syntax reverse engineering by analysis of the intrinsic structure of individual messages," in Proc. 12th USENIX Workshop Offensive Technol., Baltimore, USA, Aug, 2018.

[5] M.-M. Xiao and Y.-P. Luo, "Automatic protocol reverse engineering using grammatical inference," J. Intelligent & Fuzzy Systems, vol. 32, no. 5, pp. 3585-3594, Apr. 2017.

[6] I. Bermudez, A. Tongaonkar, M. Iliofotou, M. Mellia, and M. M. Munafo, "Automatic protocol field inference for deeper protocol understanding," in Proc. 14th IFIP Networking Conference, pp. 1-9, Toulouse, France, May. 2015.

[7] Y. Wang, N. Zhang, Y.-M. Wu, B.-B. Su, and Y.-J. Liao, "Protocol formats reverse engineering based on association rules in wireless environment," in Proc. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communication, pp. 134-141, Melbourne, Australia, July. 2013.

[8] H. Li, B. Shuai, J. Wang, and C. Tang, "Protocol reverse engineering using LDA and association analysis," in Proc. 11th International Conference on Computational Intelligence and Security, pp. 312-316, Shenzhen, China, Dec. 2015.

[9] K. Choi, Y. Son, J. Noh, H. Shin, J. Choi, and Y. Kim, "Dissecting customized protocols: automatic analysis for customized protocols based on IEEE 802.15.4," in Proc. 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 183–193, Darmstadt, Germany, July 2016.

[10] R. Ji, J. Wang, C. Tang, and R. Li, "Automatic reverse engineering of private flight control protocols of UAVs," Security and Communication Networks, vol. 2017, ID. 1308045, pp. 1-9, Jul. 2017.

[11] M. Marchetti and D. Stabili, "READ: reverse engineering of automotive data frames," IEEE Transactions on Information Forensics and Security, vol. 14, no. 14, pp. 1083-1097, Apr. 2019.

[12] H. Fereidooni, J. Classen, T. Spink, P. Patras, M. Miettinen, A.-R. Sadeghi, M. Hollick, and M. Conti, "Breaking fitness records without moving: reverse engineering and spoofing fitbit." in Proc. 20th International Symposium, Research in Attacks, Intrusions and Defenses (RAID), pp. 48-69, Atlanta, USA, Sep. 2017.

[13] M. Shevertalov and S. Mancoridis, "A reverse engineering tool for extracting protocols of networked applications," in Proc. 14th Working Conference on Reverse Engineering, pp. 229–238, Vancouver, BC, Canada, Oct. 2007.

[14] J.-Z. Luo and S.-Z. Yu, "Position-based automatic reverse engineering of network protocols," J. Network and Computer Applications, vol. 36, no. 3, pp. 1070–1077, May, 2013.

[15] I. Bermudez, A. Tongaonkar, M. Iliofotou, M. Mellia, and M. M. Munafo, "Automatic protocol field inference for deeper protocol understanding," in Proc. 14th IFIP Network Conference, pp. 1–9, Toulouse, France, May. 2015.

[16] W. Cui, J. Kannan, and H. J. Wang, "Discoverer: automatic protocol reverse engineering from network traces," in Proc. 16th USENIX Security. Symposium, pp. 199–212, Boston, MA, USA, Aug. 2007.

[17] Behrouz A. Forouzan, "TCP/IP protocol suite(4th ed.)," McGraw-Hill Higher Education, 2009.

[18] Y.-H. Goo, K.-S. Shim, M.-S. Lee, and M.-S. Kim, "Protocol specification extraction based on Contiguous Sequential Pattern algorithm," IEEE Access, vol. 7, pp. 36057-36074, Mar. 2019.

[19] G. Bossert, F. Guihery, and G. Hiet, "Towards automated protocol reverse engineering using semantics information," in Proc. 9th ACM Symposium on Information, Computer and Communications Security, pp. 51–62, Kyoto, Japan, Jun. 2014.

[20] Z. Wang, X. Jiang, W. Cui, X. Wang, and M. Grace, "ReFormat: automatic reverse engineering of encrypted messages," in Proc. 14th European Symposium on Computer Security (ESORICS), pp.200-215, Springer, Berlin, Heidelberg, Sep. 2009.

[21] J. Caballero and D. Song, "Automatic protocol reverse-engineering: message format extraction and field semantics inference," International Journal of Computer and Telecommunications Networking, vol. 57, no. 2, pp. 451–474, Feb. 2013.