Design and Implementation of a Skype Protocol Analyzer

Ray-Guang Cheng1, Po-Yao Tsai1, Ping-Chen Lin1, Chia-Liang Lian1, Tse-Han Liu1, Hsien-Cheng Chou2, Shiao-Li Tsa02, and Shanchieh Yang4

1Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taiwan
2Department of Computer Science and Information Engineering, National Taiwan University, Taiwan
3Department of Computer Science, National Chiao Tung University, Taiwan
4Department of Computer Engineering, Rochester Institute of Technology, USA

Email: crg@mail.ntust.edu.tw

Abstract—Proprietary protocols on the Internet often pose challenges for users and analysts to determine how secure the applications are or how easy private information can be accessed. This work presents a tool that processes packets generated by Skype and analyzes its proprietary implementation. The goal of the exercise is to determine how one may reverse engineer a proprietary protocol and obtain private information. The results demonstrate that a buddy-list of a target user can be obtained through the developed Skype Protocol Analyzer.

Keywords—Privacy, Skype, Protocol

I. INTRODUCTION

Voice-over-Internet Protocol (VoIP) is gaining popularity on Internet, particularly through mobile devices. Different from traditional communication in which lawful inspection is in place, VoIP can be difficult to inspect and cause security threats and privacy concerns. Among others, Skype is a commonly used proprietary VoIP service using a hybrid peer-to-peer (P2P) and client-server system. It was introduced in 2003 and acquired by Microsoft in 2011. Skype stores user information in a decentralized fashion and can seamlessly traverse across Network Address Translations (NATs) and firewalls. However, Skype is proprietary and it adopts strong encryption mechanisms, which makes it difficult to evaluate its security and privacy threats.

Existing analyses of Skype were conducted from either the protocol aspect or the encryption aspect. Baset and Schulzrinne [1] found that a Skype peer-to-peer overlay network is composed of three types of nodes, Skype client, super node, and login server. A Skype client must connect to a super node and register itself with the login server for authentication. Korczynski and Duda [2] shown that Skype multiplexes several services (e.g., VoIP calls, video conferencing, instant messaging, or file transfer) using the same ports, which amplifies the problem of identifying encrypted flows that bypass firewalls and other network policies. Several methods have been used to identify encrypted flows, including those based on the port numbers [3,4,5], ones that monitor statistical properties of flows, packets, and timings [2,3,6]. Payload-based identification methods have rarely being used for Skype analysis, because protocol experts cannot obtain recognizable payload patterns with all Skype communications encrypted [3]. This set of methods was developed primarily based on external observation, which may not be enough for accurate analysis.

Biondi and Desclaux [7] and Desclaux and Kortchinsky [8] presented pioneering cryptography studies of Skype. They showed that Skype uses RC4 encryption algorithm to obfuscate the payload of Skype packets. RC4 uses Skype’s packet ID, the CRC32 of public source IP and destination IP, and Skype initial vector (Skype IV) to generate the 80-byte RC4 key. Details of Skype’s encryption and decryption algorithms were summarized by Desclaux and Kortchinsky [8] in their presentation of Vanilla-Skype, which also found that Skype developers have modified the behavior of Skype based on the potential implementation threats indicated in [7]. Bushmanov [9] extended the work of “Vanilla-Skype” and developed an open-source Skype emulator named Skyemu [10]. Skyemu allows sending messages to Skype users by emulating the Skype 1.4 protocol session handshake. The above studies [7-10] provided a number of insights for the encryption algorithms used and traffic signatures exhibited by Skype, and, thus, formed the basis of this work.

II. IMPLEMENTATIONS

This work develops a Skype add-on module to filter Skype packets from a mixture of Skype and non-Skype packets in real time. The filtered Skype packets are decrypted, unpacked, and displayed using Wireshark. Fig. 1 gives an architectural view of the Skype add-on module developed using Lua. The users can easily read the content of a chosen Skype packet by double-clicking the packet. The Skyemu.dll was developed by modifying part of the modules in Skyemu [10]. We used ‘Expand_IV.c’ module to generate Skype RC4 key and ‘unpack-4142.c’ module to unpack and decompress the selected packet in real-time. The decompressed content of the selected packet is then displayed through Wireshark graphic user interface (GUI). Note that ‘unpack-4142.c’ was originally developed by O’Neil [11].

The Skype packet filter was developed based on the classification of Skype packets presented in [12]. We observed that the use of TCP or UDP transmission depends on functions, network environment, and operational stages. A Skype client uses UDP packets to probe the network conditions for real-time network policies. Several methods have been used to identify encrypted flows, including those based on the port numbers [3,4,5], ones that monitor statistical properties of flows, packets, and timings [2,3,6]. Payload-based identification methods have rarely being used for Skype analysis, because protocol experts cannot obtain recognizable payload patterns with all Skype communications encrypted [3]. This set of methods was developed primarily based on external observation, which may not be enough for accurate analysis.

Biondi and Desclaux [7] and Desclaux and Kortchinsky [8] presented pioneering cryptography studies of Skype. They showed that Skype uses RC4 encryption algorithm to obfuscate the payload of Skype packets. RC4 uses Skype’s packet ID, the CRC32 of public source IP and destination IP, and Skype initial vector (Skype IV) to generate the 80-byte RC4 key. Details of Skype’s encryption and decryption algorithms were summarized by Desclaux and Kortchinsky [8] in their presentation of Vanilla-Skype, which also found that Skype developers have modified the behavior of Skype based on the potential implementation threats indicated in [7]. Bushmanov [9] extended the work of “Vanilla-Skype” and developed an open-source Skype emulator named Skyemu [10]. Skyemu allows sending messages to Skype users by emulating the Skype 1.4 protocol session handshake. The above studies [7-10] provided a number of insights for the encryption algorithms used and traffic signatures exhibited by Skype, and, thus, formed the basis of this work.

Fig. 1. Architecture of the proposed Skype add-on module for Wireshark.
time transmission, and to determine if a Skype client is located behind a NAT or a firewall. A Skype client uses TCP packets to establish a connection with the super node, to communicate with the login server for authentication, to transmit text messages, and to exchange the session key. In Skype, a list of super nodes’ IP address and port numbers is stored in Host Cache, which can be obtained from the HostCache field of shared.xml under Skype folder. One can get the entire string value of the HostCache field: remove the first 18 characters of the string, segment the rest of the string value to segments of 12 and 52 characters, and convert the 12-character long segments to find a list of super nodes IP addresses and port numbers. Further observed that a Skype client uses IP 0.0.0.0 to transmit packets to an external peer node if it does not have a public IP. The external peer node will use the public IP of the NAT to decrypt the received packets and the decryption always fail because of using a wrong source IP. The external peer node will then send a NAck carrying the public IP of the Skype client. Therefore, the Skype client can retrieve its public IP from the NAck packet.

III. EXPERIMENTAL RESULTS

The developed Skype add-on module can detect all Skype traffic regardless of software version, and allows us to analyze the behavior of Skype. This tool is available for download in [13]. First, the behavior of Skype client in retrieving the Host Cache is different before and after Skype was acquired by Microsoft. The older versions of Skype client access the super nodes through a hard-coded list if the HostCache does not exist, while the newer versions, after acquired by Microsoft, always query the domain name server (DNS) to find the new HostCache.

In terms of port number usage, we considered Skype client version of 5.3.0.113. The port numbers 40001 to 40047 are used by super nodes; port number 33033 is used by login server; port number 12350 is almost always used for user setting privacy settings, online status, and language setting; port number 23456 is used by SkypeOut service; port number 443 is used by advertisement and applications such as Facebook and MSN; port number 80 is used to check Skype version and software upgrade; port number 12392 is used to check hardware capability; and port number 12351 is used for value added services.

The Skype protocol analyzer uses the module provided by Skymu to decrypt ‘RC4 encrypt’ field of the UDP probe packets. The RC4 encrypt data is composed of four parts: ‘Length,’ ‘CMD,’ ‘Skype ID,’ and ‘Data.’ The ‘Length’ field denotes the length of the packet size containing ‘Skype ID’ and ‘Data’ fields. The ‘Length’ field is 1 byte if the length of the packet size does not exceed 128 bytes. Otherwise, the field is 2 bytes. The Skype command is indicated in the ‘CMD’ field. The ‘CMD’ field has 2 to 4 bytes.

Finally, the Skype protocol analyzer can automatically generate a buddy-list as shown in Fig. 2. The Skype ID, the IP address, and the port number of the Skype clients shown on the buddy-list are automatically collected from the observed packets. The proposed protocol dissector can filter and extract Skype packets, decrypt and identify the content of Skype packets to generate the buddy-list of a target user. This study not only presents a tool to analyze the proprietary Skype implementation but also suggests the security and privacy threats of Skype.

REFERENCES


Fig. 2. A buddy-list automatically generated by Skype Protocol Analyzer.