Hindawi
Security and Communication Networks
Volume 2022, Article ID 4862571, 6 pages
https://doi.org/10.1155/2022/4862571

Research Article

Security Analysis of Shadowsocks(R) Protocol

Qingbing Ji ,[1,2] Zhihong Rao ,[1,2] Man Chen ,[2] and Jie Luo [2]

[1]*School of Cybersecurity, Northwestern Polytechnical University, Xi'an 710072, China*
[2]*No.30 Institute of CETC, Chengdu 610041, China*

Correspondence should be addressed to Qingbing Ji; jqbdxy@163.com

Received 8 November 2021; Accepted 31 December 2021; Published 19 January 2022

Academic Editor: Zhili Zhou

Shadowsocks(R) is a proxy software based on Socks5, which is the collective name of shadowsocks and shadowsocksR. Shadowsocks(R) is a private protocol without a handshake negotiation mechanism. Peng broke the confidentiality of shadowsocks by exploiting vulnerability in the shadowsocks protocol and decrypted the shadowsocks packets encrypted with none-AEAD encryption options using a redirection attack. Chen et al. started with the cryptographic algorithm used by shadowsocks(R) and preliminarily discussed the confidentiality of user data under the protection of shadowsocks(R) in theory. Based on Chen's work, this paper further clarifies the shadowsocks(R) protocol format and studies the encryption mechanism of shadowsocks(R) from the perspective of protocol analysis. The vulnerability of the shadowsocks(R) encryption mechanism is found, and an attack method of shadowsocks(R) is proposed. The attack method is a passive attack and can decrypt the shadowsocks packets encrypted with any encryption option. Compared with Peng's attack method, the method is more effective and more suitable for actual attacks. Finally, some methods to improve the protocol security of shadowsocks(R) are proposed.

1. Introduction

Shadowsocks (SS) is an open-source tool for scientific Internet access [1]. It is an encrypted transmission protocol based on the Socks5 proxy. ShadowsocksR [2] (SSR) adds some data obfuscation methods based on shadowsocks, fixes some security problems, and improves QoS priority. SS and SSR are often used to break through the great firewall (GFW) to browse blocked, obscured, or disturbed content. Shadowsocks(R) is the collective name of SS and SSR. Shadowsocks(R) has the advantages of being fast, difficult to detect, and cross-platform. It is currently the most used wall climbing software.

The reason why shadowsocks(R) are sought after by many people is mainly that its traffic concealment is strong and difficult to be detected by GFW. In recent years, there are many research studies [3–6] on the identification of shadowsocks traffic. In the real network, SS is relatively easier to identify than SSR, and the identification of SSR still needs further research.

The original design purpose of shadowsocks is to bypass GFW rather than provide security in the sense of cryptography. Therefore, the encryption protocol designed by shadowsocks is only limited to preshared key and no complete forward confidentiality. Peng [7] broke the confidentiality of shadowsocks by exploiting vulnerability in the shadowsocks protocol and decrypt the shadowsocks packets encrypted with none-AEAD ciphers using a redirection attack. The attack method is an active attack and cannot decrypt the shadowsocks packets encrypted with AEAD encryption options, so it has many limitations in actual attacks. Man et al. [8] started with the cryptographic algorithm used by shadowsocks(R) and preliminarily discussed the confidentiality of user data under the protection of shadowsocks(R) in theory. Based on Chen's work, this paper further clarifies the shadowsocks(R) protocol format, studies the encryption mechanism of shadowsocks(R) from the perspective of protocol analysis, and proposes an attack method of shadowsocks(R). The attack method is effective for all encryption options of shadowsocks(R). Finally, some methods to improve the protocol security of shadowsocks(R) are proposed to resist the existing attacks.

The structure of this paper is as follows. In Section 2, the shadowsocks(R) protocol and message format are further clarified. In Section 3, from the perspective of protocol analysis, the shadowsocks (R) encryption mechanism is analyzed in detail, its vulnerability is found, and the decryption method is given. In Section 4, according to the protocol vulnerability found, some suggestions about improving the protocol security of shadowsocks(R) are proposed. Finally, the study is summarized in Section 5.

2. Shadowsocks(R) Protocol and Message Format

Shadowsocks(R) service components include sslocal running on the local computer and ssserver running on the remote server, as shown in Figure 1 [9]. The traffic that an attacker can obtain is usually the encrypted traffic sent from sslocal to ssserver, rather than the traffic sent by the Socks5 agent of the local computer to the SS client. Therefore, the message format of encrypted traffic sent from sslocal to ssserver is analyzed here.

2.1. Shadowsocks Protocol and Message Format. The SSR client configuration interface is as shown in Figure 2. SSR mainly includes three-parameter setting areas: basic settings, protocol settings, and obfuscator settings. If you use the software only setting the parameters of the basic settings, SSR is SS at this time. As shown in Figure 2, the parts that need user configuration when using shadowsocks include server address, server port, password, and encryption. After configuration, the local SS client establishes a connection with the server port on the server address to transmit the application data processed through password and encryption.

TCP flow or UDP message is transmitted between the SS client and the server. This paper focuses on TCP flow and does not introduce the UDP message format. The TCP flow is preceded by a random number and the following bytes are ciphertext. The length of the random number is related to the encryption algorithm used. It is used as the initial vector of a cryptographic algorithm or other cryptographic parameters in different modes. The ciphertext is obtained by encrypting user data. Before the SS client forwards the user data to the SS server, the user access address will be added in front of the user data, and the address is also encrypted transmission. The formats of TCP flow and reverse TCP flow sent from the SS client to the server are shown in Figures 3 and 4, respectively.

TARGET ADDRESS in Figure 3 represents the destination address to be accessed by the user. Following the address representation method of the ocks5 protocol, TARGET ADDRESS is identified by three fields, type, destination address, and port, as shown in Figure 5.

The structure of TARGET ADDRESS is described as follows:

(1) ATYE is the abbreviation of address type. The byte size of ATYE is 1 byte. It has and only has three values: 1, 3, and 4. When ATYE = 1, the destination address type is IPv4. When ATYE = 3, destination address type is a domain name. When ATYE = 4, the destination address type is IPv6.

(2) DST.ADDR is the value of the destination address. If the type value is 1, the destination address is a 4-byte IPv4 address. The type value is 3, and the destination address is a variable-length string. The first byte represents the byte length of the next destination address. The type value is 4, and the destination address is a 16-byte IPv6 address.

(3) DST.PORT is a 2-byte port number.

Encryption (TARGET ADDRESS||USER DATA) refers to the ciphertext after encrypting the concatenated string of target address and user data using the cipher algorithm selected by the SS encryption option.

SS encryption option supports AES-128-CTR, chacha20-IETF, and other stream cipher algorithms, as well as AES-192-GCM, chacha20-IETF-poly1305, and other AEAD cryptographic algorithms. SS encryption options are selected differently, and the format of encrypted user data is also different. For the stream cipher algorithm, the data length before and after encryption is the same. For the AEAD cryptographic algorithm, the plaintext will become the ciphertext, as shown in Figure 6.

Rand num in Figure 3 is a random number, which is transmitted in plaintext. The length of the random number is different with different encryption algorithms. AES-128-CTR and other CTR terminated stream ciphers have a corresponding random number length of 16 bytes. Except that bf-CFB is 8 bytes, other CFB terminated stream ciphers, such as AES-128-CFB and Camellia-256-CFB, have a random number length of 16 bytes. The length of the cha20-IETF random number is 16 bytes; AEAD cryptographic algorithms include Chacha20-IETF-ploy1305, AES-128-GCM, AES-192-GCM, and AES-256-GCM and have a random number length equal to the key length.

2.2. Shadowsocks(R) Protocol and Message Format. As shown in Figure 2, SSR has two more setting areas than SS, such as protocol settings and obfuscator settings, including four configuration items: Protocol, Protocol Param, Obfuscator, and Obfs Param.

There are three main modes of Protocol option, including auth_AES128_MD5, auth_AES128_sha1, and auth_chain_a. There are two main modes of the Obfuscator option, including http_simple and tls1.2_ticket_auth. The Protocol Param and Obfs Param are set according to the selected protocol mode and obfuscator mode.

When the obfuscation mode is http_simple, the traffic between the client and the server is composed of disguised HTTP packets and disguised TCP packets. When the obfuscation mode is tls1.2_ticket_auth, the traffic between the client and the server is composed of disguised TLS packets and disguised TCP packets. When the confusion mode is plain, the traffic between the client and the server is composed of disguised TCP packets, and the shadowsocks(R) is shadowsocks.

2.2.1. Packet Format of a Disguised HTTP Packet. Different from the standard HTTP packets in the network, the HTTP packets in shadowsocks(R) have disguised HTTP

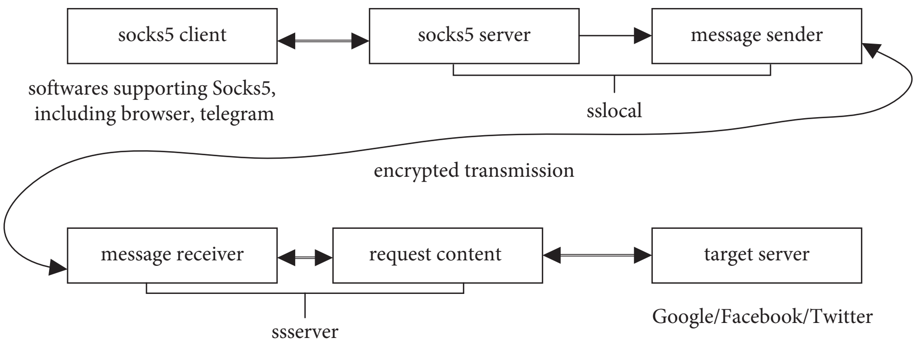
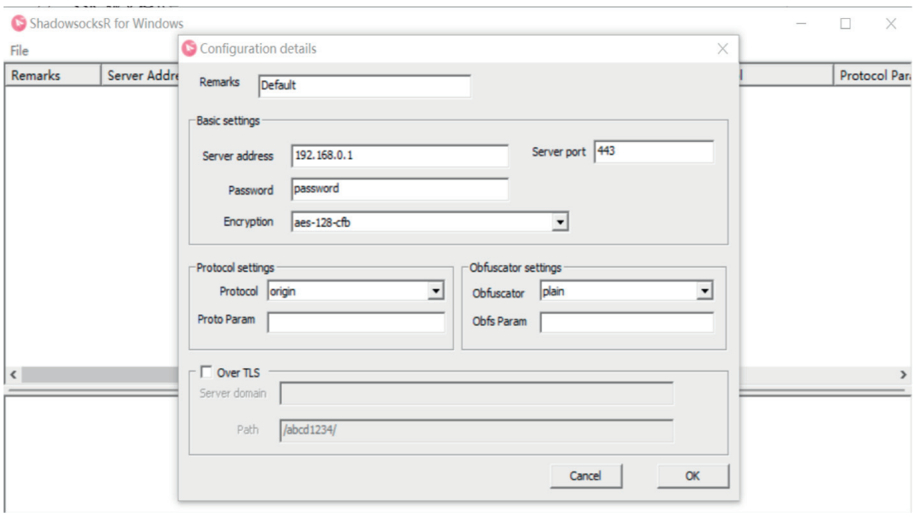


Figure 1: Communication principle of shadowsocks(R).



Remarks: Default

Basic settings

Server address: 192.168.0.1

Server port: 443

Password: password

Encryption: aes-128-cfb

Protocol settings

Protocol: origin

Proto Param:

Obfuscator settings

Obfuscator: plain

Obfs Param:

☐ Over TLS

Server domain:

Path: /abcd1234/

Figure 2: The SSR client configuration interface.

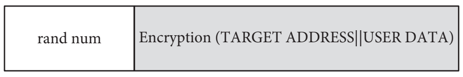


Figure 3: TCP flow from the SS client to the server.

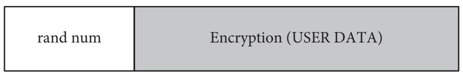


Figure 4: TCP flow from the SS server to the client.

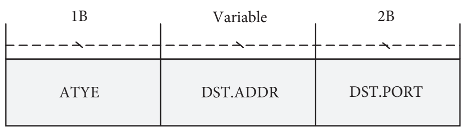


Figure 5: Structure of TARGET ADDRESS.

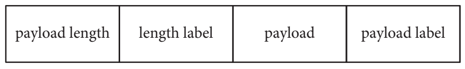


Figure 6: Ciphertext format of AEAD cryptographic algorithm.

packets and do not implement the hyperlink function. The format of the disguised HTTP packet is composed of A to F, as shown in Figure 7.

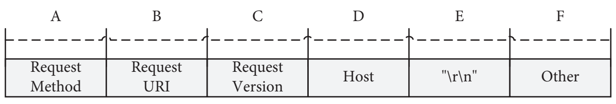


Figure 7: The format of the disguised HTTP packet.

A: according to the obfuscation mode, the value of the request method can be one of "POST/" and "GET/."

B: it consists of a URI format header and data block. The data block is filled according to the format of "%X_1%X_2...%X_N." X_1, X_2, and X_N represent the first byte, the second byte, and the Nth byte of the encrypted data.

C: request version is a fixed value. The characteristic value is HTTP/1.1.

D: host is a fixed value according to the format of "protocol parameter: server port."

E: data are a fixed value. The characteristic value is "\r\n."

2.2.2. Packet Format of Disguised TLS Packet. Different from the standard TLS packets in the network, the TLS packets in shadowsocks(R) are disguised TLS packets and do not implement the secure transmission function. The format of the TLS disguised packet is composed of A to C, as shown in Figure 8.

A: the value of the TLS flag header is selected from the following four sets of data; each set of data consists of multiple bytes shown in Table 1. ∗ means any value.

B: B represents the length of the data block, which occupies 2 bytes.

C: C represents the data block; the format is shown in Figure 9.

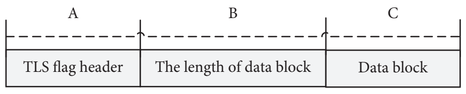


Figure 8: The format of the disguised TLS packet.

Table 1: The value of the TLS flag header.

1	0×17 0×03 0×03
2	0×16 0×03 $0\times{*}{*}$
3	0×14 $0\times0{*}$
4	$0\times{*}{*}$

3. Analysis of Shadowsocks(R) Encryption Mechanism

When the shadowsocks(R) traffic carried on TCP is known, if the shadowsocks (R) traffic can be decrypted, the confidentiality guarantee of user data will be weakened and the risk of users being supervised will increase. As we all know, the premise for an attacker to decrypt shadowsocks(R) traffic is to obtain the preshared key and encryption configuration. Next, by analyzing the encryption mechanism of shadowsocks(R), we find the method to obtain the preshared key and encryption configuration.

3.1. The Key Generation and Encryption Parameter Acquisition of Shadowsocks(R). According to the analysis of shadowsocks source code, the cryptographic algorithms used in shadowsocks include two types: key generation algorithm and encryption (decryption) algorithm used by different encryption options. When shadowsocks configures the encryption option, it essentially selects the basic key generation algorithm and encryption mode at the same time. The encryption option encrypts the data based on the random number in the TCP stream and the master key generated by the key generation algorithm. IV length is 16 bytes, and the value is the 16-byte random number of TCP flow. The value of s is 128. Different encryption options use random numbers and master keys in different ways. This paper selects a stream cipher and AEAD encryption option as examples.

3.1.1. The Basic Key Generation Algorithm. The shadowsocks key generation algorithm generates the master key based on the preshared key, that is, the password of the shadowsocks configuration interface. When the encryption option is determined, the length of the master key is determined. Currently, shadowsocks supports key lengths of 16 bytes, 24 bytes, and 32 bytes. Let pwd represent the preshared key and master key represent the master key used by the shadowsocks encryption option. Enter pwd and the length of the master key, and obtain the output master key through up to two rounds of the MD5 algorithm.

3.1.2. AES-192-CFB in Encryption Option. The prerequisite elements of CFB mode include basic cryptographic algorithm, key, supported input/output length, and supported label length [10]. The inputs include IV, plaintext, and offset parameters s. When shadowsocks uses AES-192-CFB encryption option, the basic cryptographic algorithm uses AES-192. The key length is 24 bytes, and the value is derived from the master key which generated the key generation algorithm. The length of IV is 16 bytes, the value is the 16-byte random numbers of a TCP flow, and the value of s is 128.

3.1.3. AES-256-GCM in Encryption Option. The prerequisites of GCM mode include basic cryptographic algorithm, key, supported input/output length, and supported label length [11]. The input includes IV, plaintext, and additional authentication data A. When shadowsocks uses AES-256-GCM encryption, the basic cryptographic algorithm uses AES-256; IV length is 12 bytes, and the value is 0; A length is 0 bytes. The key K is 32 bytes in length, which is generated by the value master key and the 16-byte random numbers of the TCP flow and is generated by three rounds of HMAC algorithm based on SHA-1 hash function.

3.2. The Derivation of Preshared Key and Encryption Configuration of Shadowsocks. The attacker guesses the parameters, including the preshared key and encryption configuration. Since the stream encryption algorithm lacks a verification mechanism and shadowsocks also has not designed a verification mechanism, it is possible to obtain verification information from the plaintext to verify whether the guessed parameters are correct through the principle of traversal.

The message format of the plaintext is shown in Figures 2 and 3. The payload of plaintext includes addresses and data. Shadowsocks can implement the proxy function for any application, so the attacker cannot extract useful features for verification from the data. The hostname and port number in the address are also strongly related to the user's operation and do not have obvious features for verification. However,

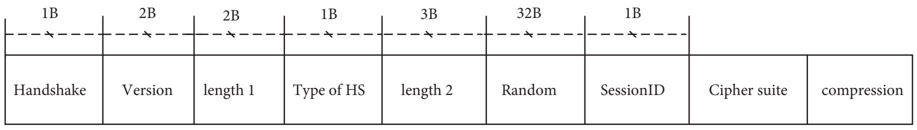


Figure 9: The format of the data block.

known as type, the first byte of address has only 3 possible values. Assuming that the guessed parameters are correct after the TCP upstream is decrypted, the probability that the first byte of address belongs to {1, 3, 4} is 1. Assuming that the guessed parameter is wrong, the probability that the first byte of address belongs to {1, 3, 4} is 3/256. For N consecutive TCP upstreams, if the guessed parameters are correct, the probability that the first byte of the plaintext belongs to the three possible values is 1. If the guessed parameter is wrong, the probability that the first byte of the plaintext belongs to the three possible values is $(3/256)^N$. When N approaches infinity, the probability approaches 0. So, as long as there is more TCP upstream traffic, the probability of collision errors will be smaller.

Based on the above analysis, an algorithm for guessing the preshared key and encryption configuration can be designed. When the first byte does not belong to {1, 3, 4}, the guessed parameter is discarded. When the guessed parameter is equal to the correct parameter, the probability of selecting the guessed parameter is 1. When the guessed parameter is not equal to the correct parameter, the probability of discarding the guessed parameter is $1-(3/256)^N$. When N approaches infinity, the probability approaches 1. In other words, the algorithm can return the correct preshared key and encryption configuration with probability 1 as long as there are enough TCP upstream. Let $N=4$; the algorithm for guessing the preshared key and encryption configuration is as follows:

Input: the first 40 bytes of TCP upstream

Output: preshared key, encryption configuration

calculation steps:

(i) Guess the preshared key.

(ii) Traverse the encryption configuration.

  (1) Determine whether the traversal of the encryption configuration is complete. If the convenience has been completed, jump to guess the next preshared key.

  (2) Determine the length of IV according to the encryption configuration guessed. Set the IV length to L.

  (3) Determine the length of the encryption key according to the encryption configuration guessed.

  (4) Input the preshared key guessed and the length of the encryption key to calculate the encryption key.

  (5) Set the countervalue to 0.

  (6) Start to traverse N TCP upstreams.

    (a) The countervalue is increased by 1.

    (b) The data1 is read from the 0th byte of the TCP upstream. The length of the data1 is L, and the data1 is set to IV.

    (c) The data2 is read from the Lth byte of the TCP upstream. The length of the data2 is M-L, and the data2 is set to D.

    (d) IV, the preshared key guessed and D are input as decryption parameters to obtain the plaintext P.

    (e) If the first byte of plaintext P does not belong to {1, 3, 4}, jump to traverse the next encryption configuration.

    (f) If the first byte of plaintext P belongs to {1, 3, 4} and the countervalue is not N, jump to the next TCP upstream.

    (g) If the first byte of plaintext P belongs to {1, 3, 4} and the countervalue is N, the preshared key and encryption configuration are output.

3.3. Correctness Verification of the Preshared Key and Encryption Configuration Based on Shadowsocks.

The destination IP address and destination port, respectively, correspond to the relay server IP and relay server port in the configuration parameters. It is possible to get the destination IP address and destination port by analyzing TCP flow based on tools such as Wireshark. If the attacker can successfully connect to the relay server by dialing back, it means that the preshared key and encryption configuration are correct. Dialing back requests the configuration parameters including server IP, server port, the preshared key, and encryption algorithm.

3.4. The Derivation of Preshared Key and Encryption Configuration of Shadowsocks(R).

According to the analysis of shadowsocks(R) source code, encryption, protocol camouflage, and data obfuscation are three relatively independent processes. Firstly, shadowsocks(R) encapsulates the user data using the protocol and protocol parameters to make the user data meet the format of the protocol. Shadowsocks(R) encrypts the data meeting the protocol format in the same way as shadowsocks. Shadowsocks(R) encapsulates the encrypted data with obfuscation and obfuscation parameters to make the encrypted data meet the obfuscated format.

The preshared key and encryption configuration acquisition method of shadowsocks(R) is similar to that of shadowsocks. Obfuscation and obfuscation parameters do not involve encryption and decryption steps, and their influence can be eliminated by directly unpacking according to the obfuscated packaging format. Due to the influence of protocol and protocol parameters, the plaintext characteristics after TCP flow decryption are randomized, and it is impossible to determine whether the parameters are correct through the plaintext characteristics. However, because each protocol uses a specific protocol format, the correctness of preshared key and encryption configuration and the correctness of protocol parameters can be determined by verifying whether the plaintext meets the protocol format. Finally, the attacker can further guess and determine the protocol parameters according to the protocol type and the plaintext obtained through preshared key, encryption, and decryption.

After obtaining all parameters, the attacker can decrypt SSR traffic by reading obfuscation and obfuscation parameters, removing obfuscation encapsulation, reading preshared key and encryption to obtain plaintext, reading protocols and protocol parameters, and extracting plaintext or further decryption.

4. Some Suggestions for Enhancing Shadowsocks(R) Protocol

As described in Section 3, the attacker can obtain user parameters by capturing network traffic and traversing parameters. Once the user parameters are obtained, the attacker can listen to all user data, seriously threatening data confidentiality. Because the attacker uses the passive attack method, the user has no perception of this attack behavior. To deal with these threats, some suggestions to enhance the security strength of SS protocol are proposed as follows:

(1) We recommend that users use shadowsocks(R) more, and must configure obfuscation parameters and protocol parameters during configuring relay server, including auth_AES128_MD5, auth_AES128_sha1, and auth_chain_a.

(2) Try to choose AEAD cryptographic algorithm instead of stream algorithm, including AES-GCM-256, AES-GCM-192, AES-GCM-128, and CHACHA20-IETF.

(3) Shadowsocks(R) is not an encryption protocol designed by the government. Therefore, the identity verification of shadowsocks(R) is limited to the preshared key, and there is no complete forward secrecy. It is recommended to add the complete forward secrecy in the upgraded version.

(4) Shadowsocks(R) use SM3 and Sha-3 to replace the existing hash algorithm and add salt value in the process of master key generation.

5. Conclusion

Shadowsocks(R) is currently the most used circumvention software. Shadowsocks(R) is not an encryption protocol designed by the government. Therefore, there are some loopholes in the encryption mechanism. This study analyzes the vulnerability of encryption mechanisms based on shadowsocks(R). Methods for obtaining the preshared key and encryption configuration are proposed. Finally, we propose some suggestions for enhancing the shadowsocks(R) protocol, which will reduce the probability of shadowsocks(R) being attacked and decrypted.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors of this paper declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Key Laboratory of Confidential Communication Fund Project (no. 6142103190308).

References

[1] https://sourceforge.net/projects/shadowsocksgui/.

[2] https://github.com/ShadowsocksR-Live/shadowsocksr.

[3] Z. Deng, Z. Liu, Z. Chen, and Y. Guo, "The random forest based detection of Shadowsock's traffic," in *Proceedings of the 9th International Conference on Intelligent Human-Machine Systems and Cybernetics*, pp. 75–78, Hangzhou, China, August 2017.

[4] H. Hangsong, "Research on shadowsocks traffic identification based on xgboost algorithm," *Software Guide*, vol. 17, no. 12, pp. 200–203, 2018.

[5] X. Zeng, X. Chen, G. Shao et al., "Flow cb based s traffic identification," *IEEE Access*, vol. 7, no. 7, pp. 41017–41032, 2019.

[6] Q. Ji, X. Deng, L. Ni, and H. Lei, "Research on ShadowsocksR traffic identification based on xgboost algorithm," *Advances in Intelligent Systems and Computing*, vol. 1304, pp. 53–61, 2021.

[7] "Redirect attack on Shadowsocks stream ciphers," 2020, http://cn-sec.com/archives/305062.html.

[8] C. Man, T. Cheng, and Q. Ji, "Safety analysis of shadowsocks and ShadowsocksR," *Communications Technology*, vol. 53, no. 5, pp. 1240–1243, 2020.

[9] "Analysis of communication principle and attack method of shadow locks," 2020, https://wonderkun.cc/2020/02/18/shadowsocks%E7%9A%84%E9%80%9A%E4%BF%A1%E5%8E%9F%E7%90%86%E4%BB%A5%E5%8F%8A%E6%94%BB%E5%87%BB%E6%96%B9%E6%B3%95%E5%88%86%E6%9E%90/.

[10] M. Dworkin, *SP800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, National Institute of Standards & Technology, Gaithersburg, MD, USA, 2001.

[11] M. Dworkin, *SP800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode(GCM) and GMAC*, National Institute of Standards & Technology, Gaithersburg, MD, USA, 2007.